DISEÑO E IMPLEMENTACION DE UN APLICATIVO SOBRE DISPOSITIVOS MOVILES PARA LA SOLUCION DE UNA ECUACION DE PRIMER GRADO, DETERMINANDO LA PENDIENTE Y GRAFICANDO EL RESULTADO

YOBBY DANNA CARVAJAL OCHOA

UNIVERSIDAD LIBRE

FACULTAD DE INGENIERIA

PROGRAMA DE INGENIERIA DE SISTEMAS

BOGOTA D. C.

FEBRERO DE 2013

AUTORES

YOBBY DANNA CARVAJAL OCHOA

PROYECTO DE GRADO PARA OPTAR AL TITULO DE INGENIERO DE SISTEMAS

DIRECTOR DEL PROYECTO

ING. NESTOR FORERO

UNIVERSIDAD LIBRE

FACULTAD DE INGENIERIA

PROGRAMA DE INGENIERIA DE SISTEMAS

BOGOTA D. C.

FEBRERO DE 2013

HONORABLE TRIBUNAL EXAMINADOR

Cumpliendo con los requisitos exigidos por la UNIVERSIDAD LIBRE presento a su consideración el trabajo nombrado

DISEÑO E IMPLEMENTACION DE UN APLICATIVO SOBRE DISPOSITIVOS MOVILES PARA LA SOLUCION DE UNA ECUACION DE PRIMER GRADO, DETERMINANDO LA PENDIENTE Y GRAFICANDO EL RESULTADO

Tema que ha sido aprobado anteriormente en la etapa de propuesta de grado y anteproyecto, este último en Febrero de 2013

YOBBY DANNA CARVAJAL OCHOA

Bogotá 14 de Febrero de 2013

AGRADECIMIENTOS

A mi madre por su apoyo y por querer siempre lo mejor para mi

A Sonia Milena Torres Caicedo por su colaboración y orientación en este trabajo

A mis docentes y al equipo que conforma la facultad de ingeniería de la Universidad Libre sede el bosque, por sus enseñanzas y por su compromiso hacia las actividades administrativas que de alguna manera facilitaron mi desempeño profesional

TABLA DE CONTENIDO

TABLA DE ILUSTRACIONES

RESUMEN

De un tiempo para acá nos ha sorprendido de forma relevante el rápido avance de los dispositivos móviles que se encuentran en el mercado y que han tenido su auge por las diversas funcionalidades que tienen en la cotidianidad de nuestras vidas, por ejemplo el fácil acceso a internet fuera del hogar de forma remota y móvil, el uso de los demás servicios que nos ofrecen estos dispositivos móviles nos enamoran, nos hacen sentirnos incapaces de dejarlos totalmente de lado, nos hacen cada vez más dependientes de ellos y tras estos dispositivos, realmente la que nos proporciona el acceso a todo tipo de contenidos, conectividad, y una variedad de servicios personalizados es la tecnología que siempre está en constante cambio y evolución.

El éxito de estas tecnologías radica en la aparición de nuevas formas de interacción, con entornos muy agradables, llamativos y vistosos ofreciendo una interacción cada vez más real y natural, todas estas características lo hacen atractivos a cualquier persona y por ende esta industria ha tenido un crecimiento acelerado y se ha posicionado muy bien a nivel empresarial, inclusive estos avances tecnológicos se han tornado de vital importancia a tal punto de responsabilizarse de unos de los mayores tesoros de cualquier Compañía como lo es la información, la seguridad, la practicidad, la eficiencia y la optimización del tiempo.

INTRODUCCION

A través de este documento se plantea paso a paso el desarrollo de este proyecto de Grado que se titula *“Solución aritmética y grafica de ecuaciones de primer grado sobre dispositivos móviles*”

En este trabajo describe la razón principal del desarrolló de este proyecto, el cómo surgió el interés de adelantar un aplicativo que diera solución sencilla a las diferentes ecuaciones de primer grado, hallará la pendiente y adicional a eso las graficara, en este camino aprendimos las diferentes plataformas y tecnologías para dispositivos móviles y el cómo desarrollarlos para que funcione muy bien y se pueda acceder a los diferentes servicios que la tecnología tiene para nosotros.

Para el desarrollo de dicha aplicación móvil se tuvieron en cuenta los protocolos correspondientes y el procedimiento que se debe tener en cuenta para una buena implementación, entre ellos la arquitectura de software, la que se utilizó fue Cliente/Servidor, la Arquitectura es un concepto indispensable dentro de las aplicaciones de hoy en día, se refiere a la combinación e información de múltiples ambientes o plataformas, las aplicaciones de negocios no están circunscritas a la base de datos o el servidor de aplicaciones, por ejemplo. Cubren todos los recursos informáticos disponibles en buen funcionamiento en una organización que requieren entonces una definición de arquitectura.

De acuerdo con Kruchten, Phili: “La arquitectura de software, tiene que ver con el diseño y la implementación de estructuras de software de alto nivel. Es el

resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales, como la confiabilidad, escalabilidad, portabilidad, y disponibilidad."

En la arquitectura Cliente servidor prima la descentralización y la distribución de responsabilidades y conocimientos el modelo cliente-servidor permite reflejar mediante un sistema informático distribuido, la naturaleza distribuida de las organizaciones. Las ventajas de un sistema distribuido son evidentes: tolerancia a fallos, alta disponibilidad de la información local, menor coste global, más falibilidad para escalar el software o el hardware, etc. Sin embargo, desde el punto de vista de desarrollo de aplicaciones surgen nuevos conceptos y necesidades, y se requiere alguna forma de estructurar los desarrollos. El modelo cliente servidor permite dotar a las aplicaciones de estructura y hacer abstracción de determinados detalles de bajo nivel (aspectos de interconexión).

1. PLANTEAMIENTO DEL PROBLEMA

La revolución digital de los dispositivos móviles en la actualidad ha transformado la estructura de los medios de comunicación, la cultura y en consecuencia las industrias afines, contribuyendo a la globalización de la información donde cada ser humano es su destinatario natural, creando un desafío a nivel individual empresarial tecnológico ya que estas están sometidas a un régimen mercantil, por tanto el mundo académico tampoco está ajeno a estas presiones de avances tecnológicos que pueden beneficiar a docentes, estudiantes y diferentes profesionales.

Una característica peculiar que surge con la masificación de los dispositivos móviles, es la oportunidad que este aspecto tecnológico le brinda a las empresas para romper la brecha digital, compenetrar a sus usuarios comunes y llegar a nuevos usuarios. Las compañías que prestan estos servicios al notar la gran demanda se ven obligadas a modernizar sus servicios para realizar mejor su labor, aunque de allí surge el verdadero problema con estos, ya que las compañías se esmeran en crear nuevas aplicaciones que a la larga la mayoría son innecesarias en sentido estricto de la utilidad, Por otro lado, en la actualidad existe una diversa gama de fabricantes de dispositivos móviles, cada uno de los cuales facilita a los usuarios una interfaz para su uso basado en un sistema operativo (OS de sus siglas en inglés Operative System) definido por el fabricante. Como consecuencia, al no existir un estándar y/o metodología abierta de desarrollo que tome en cuenta

estos dispositivos, la construcción de aplicaciones para dispositivos móviles está estrechamente relacionada con el OS particular de cada fabricante.

En este sentido, las organizaciones especializadas en construir aplicaciones para los dispositivos móviles, se enfrentan a un entorno de diversas plataformas de desarrollo, lo cual les obliga a tener que desarrollar la misma aplicación para distintos OSs. Esto representa un coste para las corporaciones de tecnología móvil en cuanto a la inversión de tiempo para el aprendizaje de cada una de las plataformas de desarrollo en cuanto a su funcionalidad, sintaxis, depuración, realización de pruebas, Complicaciones por desconocimiento del riesgo tecnológico a la hora de realizar la planificación de las etapas de la construcción del software.

Adicionalmente, esta problemática se complica aún más con el pasar del tiempo y el avance de la tecnología relacionada con este tipo de dispositivos que hace que, la aparición de nuevos sistemas operativos y la evolución de los ya existentes, implique nuevos desarrollos o ajustes en los desarrollos ya finalizados.

Basado en lo anterior expuesto, sería deseable entonces contar con un ambiente de desarrollo que permita, a los desarrolladores de aplicaciones para dispositivos móviles, hacer transparente el OS específico de cada fabricante y, más aún, contar con elementos de interfaz abierta que permita hacer también transparente la incorporación de nuevas tecnologías o nuevas versiones de tecnologías existentes.

2. FORMULACIÒN DEL PROBLEMA

El principal obstáculo para el estudiante es la poca interoperabilidad existente entre, la variedad de plataformas disponibles para dispositivos móviles. El desarrollador se ve obligado a particularizar su servicio a una sola plataforma, que permita llevar a cabo soluciones aritméticas y gráficos de ecuaciones por el término de mayor grado en dispositivos móviles a los que estudiantes, docentes y demás usuarios puedan tener fácil acceso, dado a que las operaciones de carga, ejecución, prueba y depuración de programas en dispositivos móviles resultan muy laboriosas aun disponiendo de emuladores; una aplicación sencilla puede demandar tiempo para su implementación.

Dado al anterior planteamiento y teniendo en cuenta que en la actualidad cualquier persona tiene fácil acceso a dispositivos móviles y medios electrónicos, entre ellos estudiantes, docentes y público en general, se pretende facilitar la solución de ecuaciones aritméticas y graficarlas de acuerdo al primer, segundo y tercer grado sobre dispositivos móviles.

3. JUSTIFICACIÒN

La inconformidad que existe entre los diferentes usuarios por la programación que contienen los diferentes dispositivos móviles se ha vuelto un problema más común de lo que se piensa.

Durante mucho tiempo las personas han utilizado estos dispositivos para comunicarse o como una herramienta de trabajo pero no existe uno de estos que este programado especialmente para un grupo de personas o para un campo laboral especifico como los estudiantes y profesores, que les facilite desarrollar soluciones aritméticas, graficar ecuaciones de primer, segundo y tercer grado en sus dispositivos móviles donde se muestre el resultado de estas operaciones de manera rápida y confiable, facilitando así el desempeño académico de los usuarios y sobretodo que no tengan que preocuparse por los asuntos de compatibilidad entre las diferentes plataformas que existen hoy en día en el mercado.

4. OBJETIVOS

4.1 OBJETIVO GENERAL

Desarrollar un software aplicativo para dispositivos móviles bajo el lenguaje de programación Java ME y Java SE, con el objeto de estandarizar y adaptar diferentes plataformas relacionadas con los sistemas operativos asociados a los fabricantes de estos dispositivos, a fin de simplificar el proceso de gráficas y resolver ecuaciones de primer grado, en donde solo se visualizará el resultado de la ecuación, hallar la pendiente y realizar su respectiva gráfica.

4.2 OBJETIVOS ESPECIFICOS

- Estudiar y analizar los antecedentes sobre la elaboración de entornos de desarrollo multiplataforma para aplicaciones matemáticas.
- Diseñar un programa de estandarización en el desarrollo de aplicaciones para dispositivos móviles bajo el lenguaje de programación Java ME y Java SE.
- Analizar el funcionamiento de entornos de desarrollo basado en diversidad de sistemas operativos de los dispositivos móviles.
- Investigar y analizar, los sistemas operativos más usados presentes en el mercado de los dispositivos móviles.
- Recrear un escenario de simulación para evaluar la propuesta de diseño que contiene esta plataforma.
- Analizar los resultados de la evaluación y formular las conclusiones pertinentes.

5. ALCANCE DEL PROYECTO

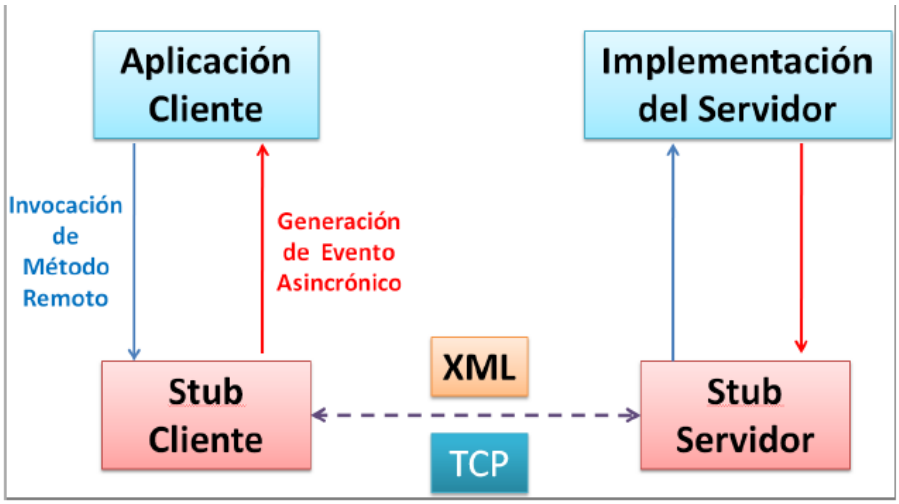
El Análisis y desarrollo de este estudio será lograr identificar los aspectos comunes involucrados en el desarrollo de aplicaciones para los diferentes tipos de dispositivos móviles con la complejidad de sus plataformas y la forma de estandarizar sus variables que tienen actualmente, así como también, podemos tener una idea del número de persona a los cuales les está afectando el problema y cuántos de ellos se pueden estar beneficiando con un aplicativo que resuelva ecuaciones en sus celulares así como también el número de estos con las capacidades técnicas para ejecutar este aplicativo.

6. DISEÑO METODOLÓGICO

6.1 IMPLEMENTACION

Para este proyecto lo primero que se determinó fue la arquitectura que se iba a aplicar, la más indicada para este proyecto es Cliente – servidor, en esta se trata de una capa que sirve de intermediario entre la aplicación y la capa de transporte, de forma que el flujo de datos codificados a través de la red es completamente transparente para las implementaciones del Cliente y del Servidor.

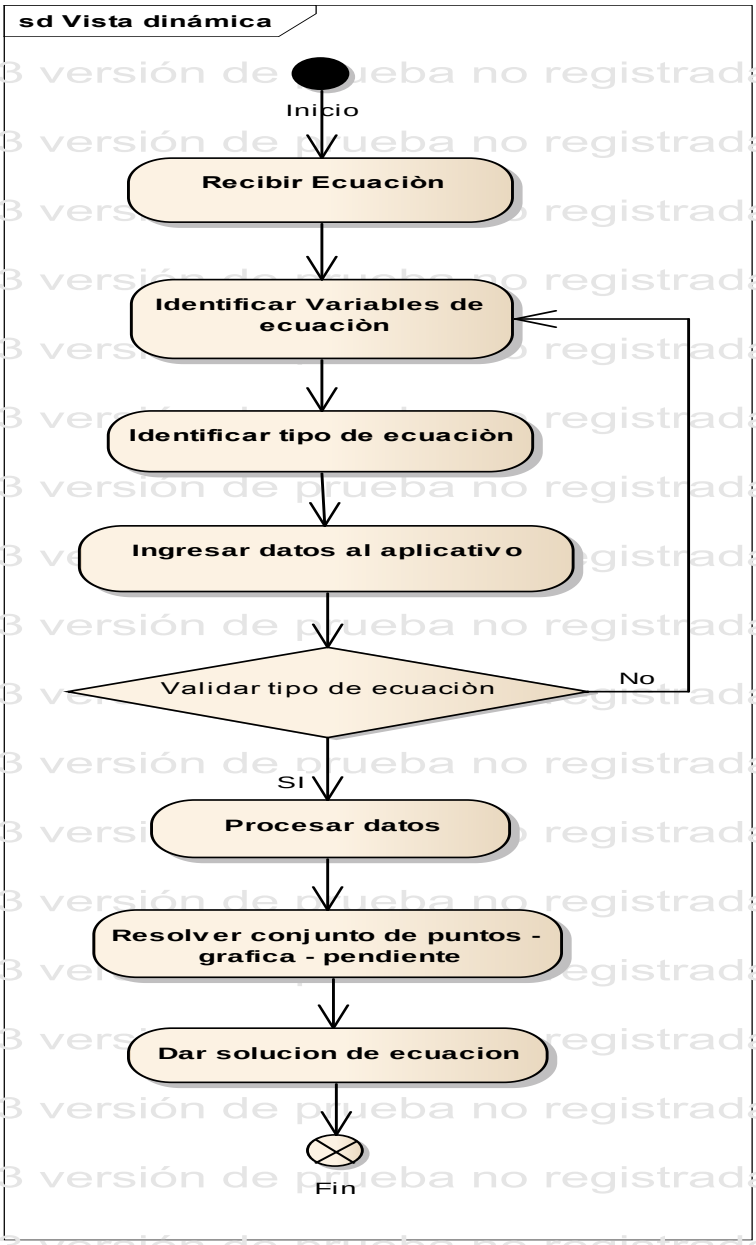
Ilustración 1. Diseño de la implementación de Cliente Servidor



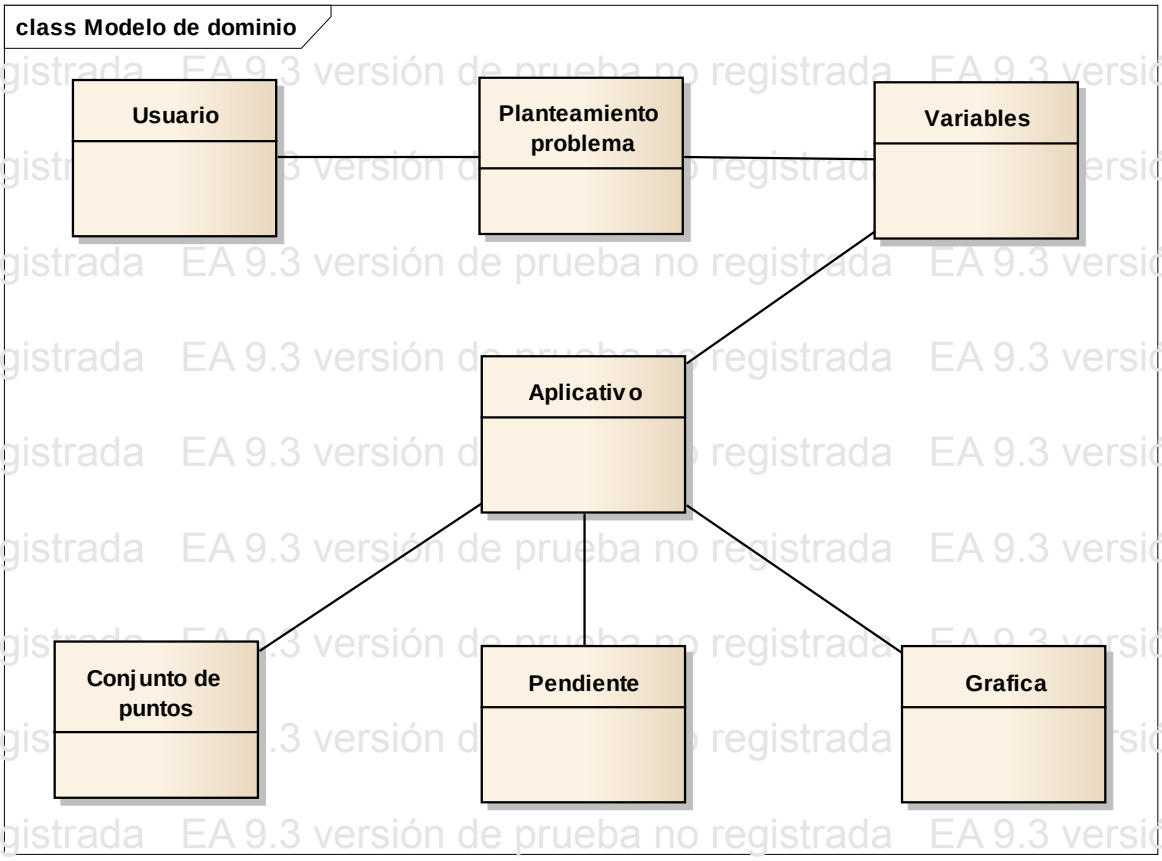
6.2 METODOLOGIA RUP

Análisis de referentes teóricos y empíricos entorno al problema que se desea desarrollar e implementar.

6.2.1. Modelamiento del Negocio



6.2.2. Modelamiento del Dominio



6.2.3. Glosario de términos

Usuario: Es el actor que recibe el problema (ecuación lineal de primer grado) del exterior e intenta resolverla mediante el aplicativo.

Planteamiento del problema: Hace referencia al problema, en este caso la ecuación de primer grado que debe resolverse a través del aplicativo.

Variables: Factores de gran influencia en el proceso de solución del problema y cuyos valores son modificables.

Aplicativo: Es el middelware en tiempo real del proceso, este soporta las peticiones sensibles al tiempo por parte del usuario, es el software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, software, redes, hardware, y/o sistemas operativos.

Conjunto de Puntos: Cada punto (x, y) que pertenece a una recta se puede representar en un sistema de coordenadas, siendo (X) el valor de la abscisa y (Y) el valor de la ordenada.

Pendiente: Es el grado de inclinación de una recta, la razón de cambio en Y con respecto al cambio en X, y se halla con el conjunto de puntos que determina el aplicativo de la ecuación propuesta.

Gráfica: Es el conjunto de puntos que resultantes de la ecuación problema, que se plasman como coordenadas cartesianas y se representan los valores en dos ejes cartesianos ortogonales entre sí.

6.2.4. Requerimientos

6.2.4.1. Funcionales

Definir Actores: Las personas encargadas de manipular el aplicativo móvil.

Usuario: Persona encargada de interactuar con el software

Lista Inicial de casos de uso

1. Usuario

- Recibir ecuación

- Identificar Variables

- Identificar tipo de ecuación

- Ingresar datos al aplicativo

- Validar tipo de ecuación

- Dar solución de ecuación

6.2.4.2. No Funcionales

Los requisitos que el aplicativo necesita para funcionar son:

Servidor

- Software
- Lenguaje de Programación para dispositivos móviles que soporte los

Sistemas Operativos mencionados anteriormente, como lo es Java ME y Java SE.

- Hardware
- El servidor utilizará dos servidores, uno redirector y otro director

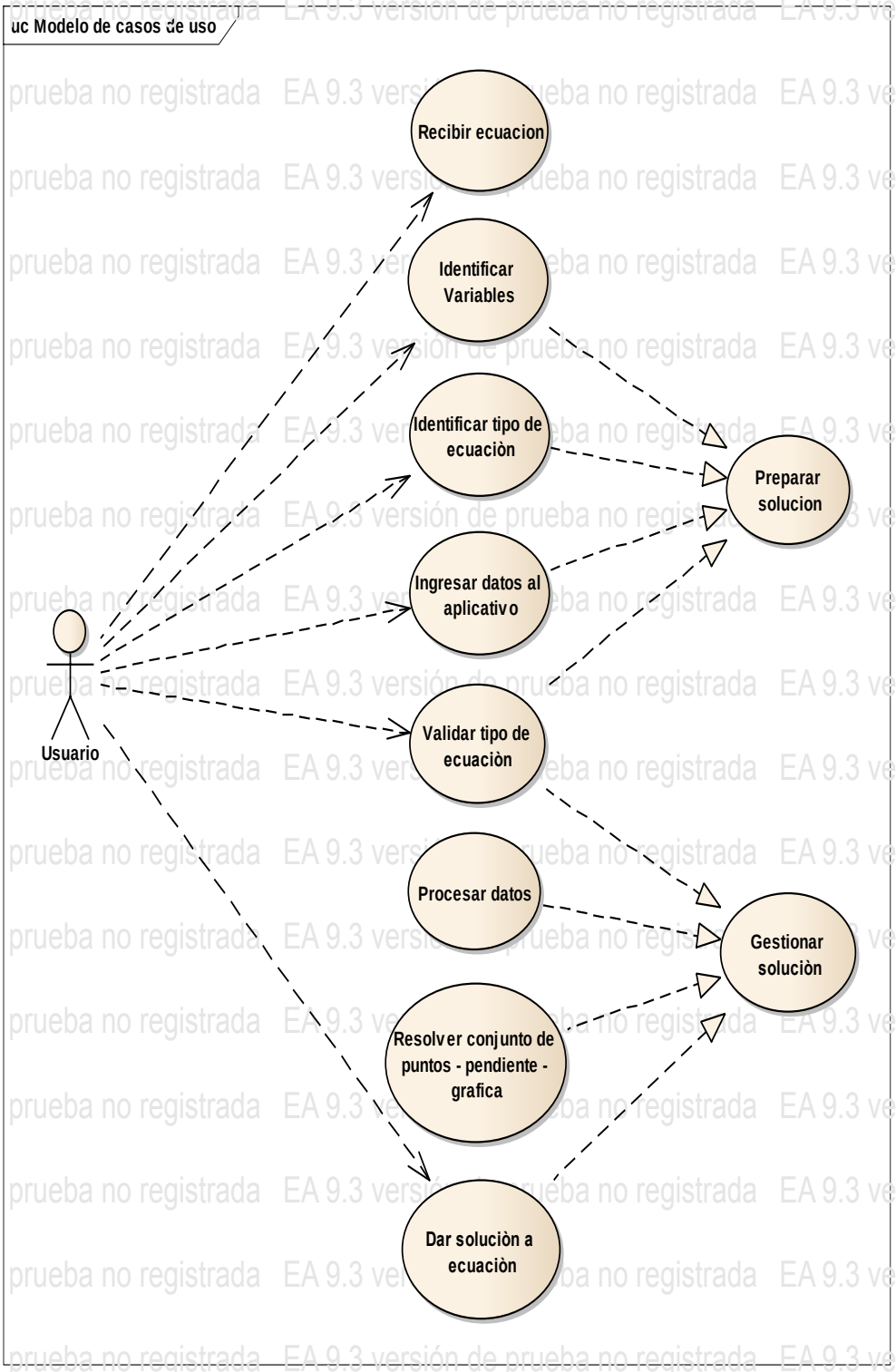
El primero interactúa como un registro, y es el que espera continuamente la conexión de clientes para redireccionarlos al servidor que presta el servicio solicitado, y el segundo servidor es donde se encuentra el servicio propiamente dicho.

Cliente

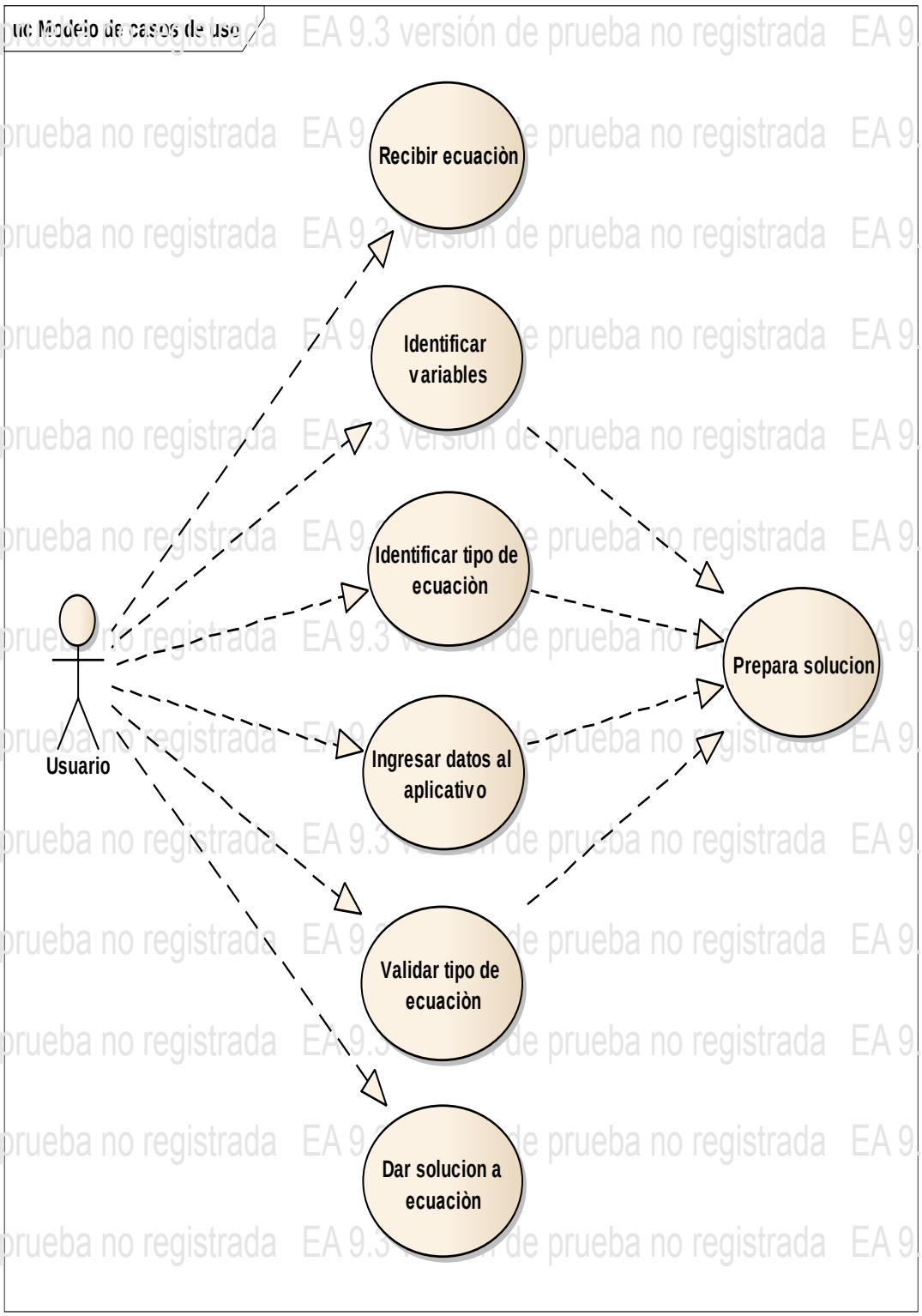
- Software
- Sistema operativo para dispositivos móviles (Google Android OS, Rim BlackBerry OS, Microsoft Windows Mobile, Symbian), excepto para dispositivos iPhone.
- Hardware
- Middleware, es el dispositivo móvil en este caso el celular que cumpla con los requisitos expuestos anteriormente y que brinde al programa una plataforma determinada que pueda interactuar con las otras implementaciones del mismo, por tal motivo la especificación del middleware es completamente independiente de la plataforma a implementar.
- El sistema operativo del dispositivo móvil

6.3. MODELOS DE CASOS DE USO

6.3.1. Diagrama general del sistema



6.3.2. Diagrama de casos de uso (Usuario)



6.3.3. Documentación de cada caso de uso

Código: 01	**Nombre: Recibir Ecuación problema**	**Periodicidad aleatorio**
Actores: Usuario		
Objetivo: Recibir del exterior (de cualquier fuente) la ecuación problema a desarrollar.		
Precondiciones: • Tener conocimiento de las ecuaciones • Diferenciar una ecuación lineal de primer grado • Retener la ecuación para posterior desarrollo		
Poscondiciones: • Querer obtener un resultado de conjunto de puntos de la ecuación problema • Querer observar la gráfica de la ecuación de acuerdo al conjunto de puntos obtenidos • Desear obtener la pendiente según los resultados mencionados anteriormente		
Curso Normal de Eventos		
Acción del actor	**Respuesta del Sistema**	
Obtener de cualquier situación o evento una ecuación para someterla a desarrollo en el aplicativo y analizar dicha situación con los resultados obtenidos.	Captura de los datos, encriptamiento de los mismos y envío de la información a un Script encargado de resolver la ecuación.	
Manejo de situaciones excepcionales: • No tener mucho conocimiento de las ecuaciones de primer grado • No saber que ecuación será el problema a resolver		

Código: 02	**Nombre: Identificar variables de la Ecuación problema**	**Periodicidad aleatorio**
Actores: Usuario		
Objetivo: El usuario recibe la ecuación problema y posteriormente para su solución identifica sus variables		
Precondiciones: • Tener conocimiento de las ecuaciones • Diferenciar una ecuación lineal de primer grado		
Poscondiciones: • Poder identificar qué tipo de ecuación de primer grado es para solucionarla con el aplicativo		
Curso Normal de Eventos		
Acción del actor	**Respuesta del Sistema**	
Obtener de cualquier situación o evento una ecuación e identifica sus variables.	Captura los datos, pide que tipo de ecuación es para proceder con la solución.	
Manejo de situaciones excepcionales: • No tener mucho conocimiento de las ecuaciones de primer grado • No saber que ecuación será el problema a resolver		

Código: 03	Nombre: Identificar el tipo de ecuación	Periodicidad aleatorio
Actores: Usuario		
Objetivo: Después de identificar las variables de cada ecuación o problema propuesto se busca situarla dentro de un tipo de ecuación y así poder desarrollarla ingresarla al aplicativo en la sección para su solución		
Precondiciones: • Retener la ecuación para posterior desarrollo • Tener conocimiento de las ecuaciones • Diferenciar una ecuación lineal de primer grado • Conocer los tipos de ecuaciones de primer grado que existen		
Poscondiciones: • Poder clasificarla entre los tipos de ecuaciones para su solución • Querer obtener un resultado de conjunto de puntos de la ecuación problema • Querer observar la gráfica de la ecuación de acuerdo al conjunto de puntos obtenidos • Desear obtener la pendiente según los resultados mencionados anteriormente		
Curso Normal de Eventos		
Acción del actor	**Respuesta del Sistema**	
Obtener de cualquier situación o evento una ecuación para identificar el tipo de ecuación lineal que es y así simplificar su solución con el aplicativo que se implementará	Captura de los datos, encriptamiento de los mismos y envío de la información a un Script encargado de mostrar los tipos de ecuaciones que hay y en donde podría ubicarse la ecuación para su posible solución.	
Manejo de situaciones excepcionales: Si el servidor Web no se encuentran funcionando ni el dispositivo móvil, estos se deben reiniciar		

Código: 04	Nombre: Ingresar datos al aplicativo	Periodicidad aleatorio
Actores: Usuario		
Objetivo: El usuario determina qué tipo de ecuación es y decide ingresar los datos bajo esta característica para obtener la solución de la gráfica		
Precondiciones: • Tener conocimiento de las ecuaciones • Diferenciar una ecuación lineal de primer grado • Seleccionar el tipo de ecuación para así introducir los datos siguiendo estas características		
Poscondiciones: Se verifica si la tipología de ecuaciones era la correcta para el desarrollo de ecuaciones lineales		
Curso Normal de Eventos		
Acción del actor	**Respuesta del Sistema**	
Obtener de cualquier situación o evento una ecuación e identifica sus variables y posteriormente ingresarlas para obtener su solución	Captura los datos, se identificar tipologías de ecuación, introducir los datos	
Manejo de situaciones excepcionales: Que la ecuación no sea de la tipología que se esperaría.		

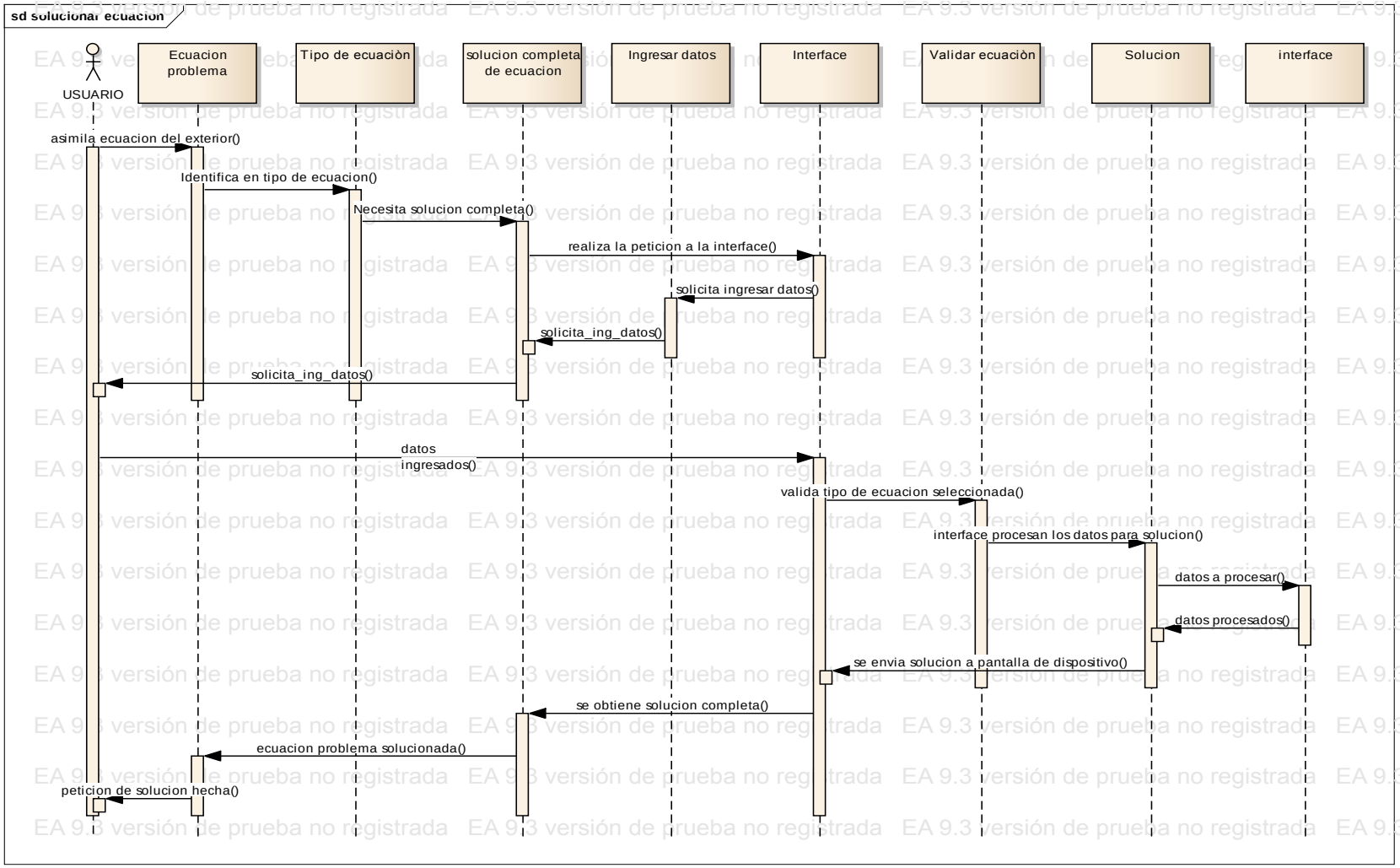
<table>
<tr><td>Código: 05</td><td>Nombre: Validar tipos de ecuación</td><td>Periodicidad aleatorio</td></tr>
<tr><td colspan="3">Actores: Usuario</td></tr>
<tr><td colspan="3">Objetivo: en este proceso es donde después de haber identificado el tipo de ecuación e haber ingresado los datos por el usuario al aplicativo, se verificar si el proceso de la identificación de ecuaciones es correcto o no y de no serlo se vuelve a realizar.</td></tr>
<tr><td colspan="3">Precondiciones:
Tener conocimiento de las ecuaciones
Diferenciar una ecuación lineal de primer grado
Seleccionar el tipo de ecuación para así introducir los datos siguiendo estas características
Validar tipo de ecuación</td></tr>
<tr><td colspan="3">Poscondiciones:
Se verifica si la tipología de ecuaciones era la correcta para el desarrollo de las ecuaciones
De estar correcta el usuario debe esperar el resultado en la pantalla</td></tr>
<tr><td colspan="3">Curso Normal de Eventos</td></tr>
<tr><td colspan="2">Acción del actor</td><td>Respuesta del Sistema</td></tr>
<tr><td colspan="2">Después de haber ingresado los datos, el usuario debe esperar un aviso de confirmación para visualizar en pantalla el resultado</td><td>Captura los datos, se identifican tipologías de ecuación, introducir los datos y el sistema si está habilitado continua con el proceso de solución, sino es así vuelve a identificarlos</td></tr>
<tr><td colspan="3">Manejo de situaciones excepcionales:
No tener claridad en los tipos de ecuaciones
Que la ecuación no sea de la tipología esperada
Si el aplicativo o el dispositivo móvil se traba y no se encuentra funcionando, estos se deben reiniciar</td></tr>
</table>

<table>
<tr><td>Código: 06</td><td>Nombre: Procesar datos</td><td>Periodicidad aleatorio</td></tr>
<tr><td colspan="3">Actores: Usuario</td></tr>
<tr><td colspan="3">Objetivo: El usuario recibe la ecuación problema, ingresa los datos en el aplicativo y luego valida el tipo de ecuación para dar una solución efectiva</td></tr>
<tr><td colspan="3">Precondiciones:
Ingresar datos al aplicativo
Valida si es una ecuación lineal de primer grado</td></tr>
<tr><td colspan="3">Poscondiciones: Dar solución a la ecuación.</td></tr>
<tr><td colspan="3">Curso Normal de Eventos</td></tr>
<tr><td colspan="2">Acción del actor</td><td>Respuesta del Sistema</td></tr>
<tr><td colspan="2">l actor luego de recibir la ecuación problema, de identificar el tipo de ecuaciones e ingresar los datos, finalmente espera el resultado de la ecuación procesada en el aplicativo.</td><td>Recibe los datos ingresados, los procesa y da una solución al actor del problema</td></tr>
<tr><td colspan="3">Manejo de situaciones excepcionales: El tipo de ecuación de primer grado no fue bien identificada de acuerdo a los conceptos teóricos, el actor debe volver a diferenciar la variable e ingresar los datos según el tipo de ecuación planteada.</td></tr>
</table>

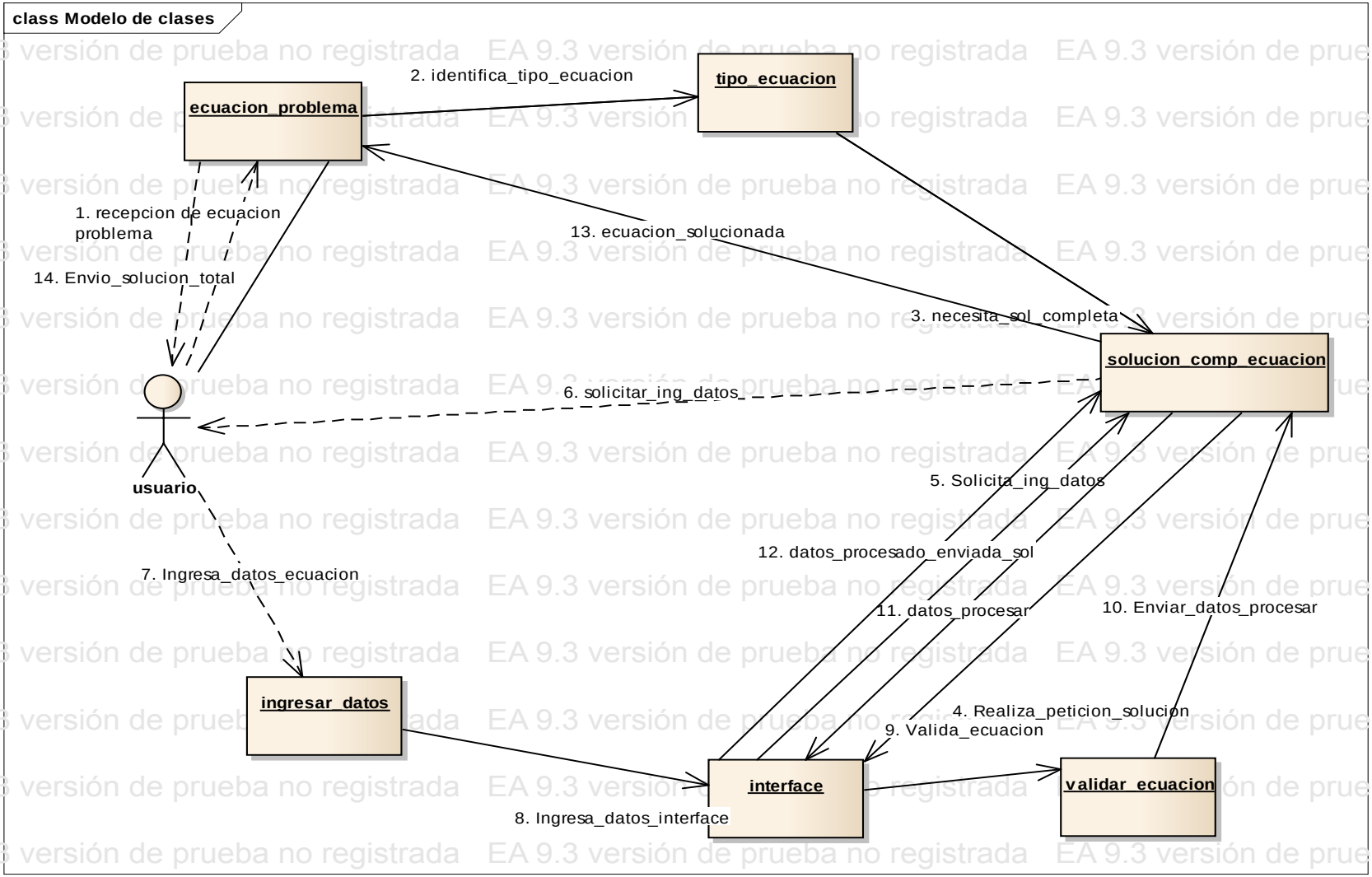
Código: 08	Nombre: Dar Solución a ecuación	Periodicidad aleatorio
Actores: Usuario		
Objetivo: El usuario debe relacionar el resultado arrojado por el aplicativo con los conocimientos teóricos previos que tiene.		
Precondiciones: • Procesa los datos • Entrega el conjunto de puntos, la pendiente y la gráfica		
Poscondiciones: Dar solución a la ecuación problema que se recibió del exterior		
Curso Normal de Eventos		
Acción del actor	**Respuesta del Sistema**	
El actor luego de recibir la solución del problema debe relacionarla con la actividad que hizo que buscar la solución de este problema.	Por pantalla muestra conjunto de puntos, la pendiente y la gráfica para entregar la solicitud del actor.	
Manejo de situaciones excepcionales: El tipo de ecuación de primer grado no fue bien identificada de acuerdo a los conceptos teóricos, el actor debe volver a diferenciar la variable e ingresar los datos según el tipo de ecuación planteada.		

Código: 07	Nombre: Entregar conjunto de puntos, la pendiente y la gráfica.	Periodicidad aleatorio
Actores: Usuario		
Objetivo: Este proceso busca entregar un resultado completo de la solución de una ecuación de primer grado, a través de una aplicación desarrollada sobre dispositivos móviles.		
Precondiciones: • Diferenciar una ecuación lineal de primer grado • Seleccionar el tipo de ecuación para así introducir los datos siguiendo estas características • Validar tipo de ecuación • Procesar los datos ingresados		
Poscondiciones: • Se verifica si la tipología de ecuaciones era la correcta para el desarrollo de las ecuaciones • De estar correcta el usuario debe esperar el resultado en la pantalla • Si no lo está debe volver a clasificar dicha ecuación.		
Curso Normal de Eventos		
Acción del actor	**Respuesta del Sistema**	
El actor de acuerdo a sus conocimientos relaciona el resultado entregado por el aplicativo con sus conocimientos teóricos previos.	El aplicativo luego de procesar esos datos, muestra el resultado en pantalla al actor, indicando el conjunto de puntos, la pendiente y la gráfica.	
Manejo de situaciones excepcionales: • Que la ecuación no sea de la tipología esperada • Si el aplicativo o el dispositivo móvil se traba y no se encuentra funcionando, estos se deben reiniciar		

6.3.4. Diagrama de secuencia



6.3.5. Diagrama de Colaboración



7. MARCO REFERENCIAL

7.1 MARCO TEORICO

7.1.1. Dispositivos móviles

Los dispositivos móviles son aparatos micro tecnológicos, con algunas capacidades de procesamiento, con tipos de conexión permanente o intermitente a una red, con memoria limitada, diseñados específicamente para una función, pero que pueden llevar consigo diferentes tipos de aplicaciones más generales y con un rol de solo entretenimiento.

También estos dispositivos se han convertido en equipos robustos para su utilización en la gestión de situaciones empresariales. Actualmente son usados para archivar una variedad de tareas y para incrementar la eficiencia, la digitalización de notas, gestión de archivos, capturas de firmas y escaneo de partes de código de barras, etc. Los dispositivos portátiles usados en el trabajo, han moldeado a través del tiempo una variedad de factores y de formas incluyendo teléfonos inteligentes en el extremo inferior, dispositivos portátiles, PDA, PC Ultra Móvil, tabletas, e incluso computadoras portátiles.

En el caso del móvil, el entorno es cambiante y dinámico. El usuario puede estar distraído o tener prisa, por lo que la forma de navegación tiene que ser realmente sencilla. Algunas formas de utilización de estos dispositivos por ejemplo, en la

vida cotidiana es comprar las entradas de cine apresurada mente desde su móvil antes de llegar a este. También son frecuentes las consultas de saldo de una cuenta bancaria antes de realizar un compra o retirar dinero y no perder el tiempo desplazándose hasta el cajero automático. En ambos caso el usuario tiene mucha prisa y se encuentra en un lugar público.

Algunas características que afectan negativamente la usabilidad de una aplicación móvil como el

Tamaño de la pantalla y el tamaño de las teclas, dificultad para escribir texto, el ancho de banda e inestabilidad de la conexión.[1]

7.1.2. Categorías de dispositivos móviles

Según el nivel de funcionalidad asociado con dispositivos móviles, en el 2005, T38 y DuPont Global Mobility Innovation Team propusieron los siguientes estándares para la definición de dispositivos móviles:[2]

Dispositivo móvil de datos limitado (Limited Data Mobile Device): dispositivos que tienen una pantalla pequeña, principalmente basada en pantalla de tipo texto con

[1] http://es.wikipedia.org/wiki/Usabilidad
http://www.webtaller.com/maletin/articulos/usabilidad_enaplicaciones_para_telefonos_moviles-4.php

[2] http://es.wikipedia.org/wiki/Dispositivo_m%C3%B3vil#Categor.C3.ADas_de_dispositivos_m.C3.B3viles

servicios de datos generalmente limitados a SMS y acceso WAP. Un típico ejemplo de este tipo de dispositivos son los teléfonos móviles.

Dispositivo móvil de datos básico (Basic Data Mobile Device): dispositivos que tienen una pantalla de mediano tamaño, (entre 30 x 120 y 240 x 240 píxeles), menú o navegación basada en íconos por medio de una «rueda» o cursor, y que ofrecen correo electrónico, lista de direcciones, SMS, y un navegador web básico. Un típico ejemplo de este tipo de dispositivos son los BlackBerry y los teléfonos inteligentes.

Dispositivo móvil de datos mejorado (Enhanced Data Mobile Device): dispositivos que tienen pantallas de medianas a grandes (por encima de los 240 x 120 píxeles), navegación de tipo stylus, y que ofrecen las mismas características que el dispositivo móvil de datos básicos más aplicaciones nativas como aplicaciones de Microsoft Office Mobile (Word, Excel, PowerPoint) y aplicaciones corporativas usuales, en versión móvil, como Sap, portales intranet, etc. Este tipo de dispositivos incluyen los sistemas operativos como Windows Mobile 2003 o versión 5, como en las Pocket PC.HOLA

7.1.3. Tipos de dispositivos móviles

- Paginadores.

- Comunicadores de bolsillo.
- Teléfonos con pantalla para Internet (Internet Screen Phones).
- Sistemas de navegación de automóviles.
- Sistemas de entretenimiento.
- Sistemas de televisión e Internet (WebTV).
- Teléfonos móviles.
- Organizadores y asistentes personales digitales (Personal Digital Assistant).

7.1.4. Sistemas operativos para dispositivos móviles

Un sistema operativo móvil (SO móvil) es el que controla un dispositivo móvil, están orientados a la conectividad inalámbrica, los formatos multimedia para móviles y las diferentes maneras de introducir información en ellos.

A medida que los teléfonos móviles crecen en popularidad, los sistemas operativos con los que funcionan adquieren mayor importancia. La cuota de mercado de sistemas operativos móviles en la actualidad es el siguiente:

7.1.5. Tipos de sistemas operativos para celulares

7.1.5.1. Android

Es un sistema operativo móvil basado en Linux, de código abierto desarrollado por Google, que busca ser fácilmente adaptable a las necesidades de cada fabricante

de dispositivos. El usuario puede interactuar con el sistema por medio de una pantalla sensible al tacto o un trackball. Tiene aplicaciones middleware, lo que lo hace ser utilizado en dispositivos móviles como teléfonos inteligentes, tabletas, Google TV y otros dispositivos. Es desarrollado por la Open Handset Alliance , un conglomerado de fabricantes y desarrolladores de hardware, software y operadores de servicio, la cual es liderada por Google y por lo general sus incorpora sus aplicaciones.

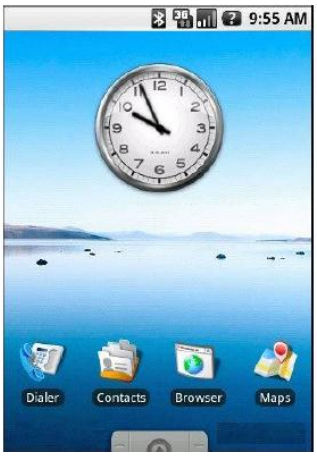
La base de este sistema es Kernel de Linux, que ejecuta a su vez varias máquinas virtuales Java para las aplicaciones del teléfono, Cada aplicación corre sobre su propia instancia de máquina virtual y cada máquina virtual es un proceso en el sistema operativo.

Sorprendentemente, esta plataforma no utiliza el estándar Java ME. Implementa la mayoría del estándar Java SE complementado con un API desarrollado especialmente para Android que permite al desarrollador un control muy profundo del sistema.

Android proporciona un emulador, un SDK, y un plugin de eclipse para facilitar la implementación de aplicaciones.[3]

[3] http://www.javeriana.edu.co/biblos/tesis/ingenieria/tesis270.pdf (pág. 41)

Ilustración 2. Visualización de celular con sistema operativo Android



7.1.5.2. Apple iphone OS

iOS (anteriormente denominado iPhone OS) es un sistema operativo móvil de Apple. Originalmente desarrollado para el iPhone, siendo después usado en dispositivos como el iPod Touch, iPad y el Apple TV. Apple, Inc. no permite la instalación de iOS en hardware de terceros. La interfaz de usuario de iOS está basada en el concepto de la tecnología multitáctil. Los elementos de control consisten en deslizadores, interruptores y botones.

La respuesta a las órdenes del usuario es inmediata y provee de una interfaz fluida. La interacción con el sistema operativo incluye gestos como deslices, toques, pellizcos, los cuales la mayoría están editadas en Estados Unidos y Reino Unido les tienen definiciones diferentes dependiendo del contexto de la interfaz. Se utilizan acelerómetros internos para hacer que algunas aplicaciones respondan a sacudir el dispositivo (por ejemplo, para el comando deshacer) o rotarlo en tres

dimensiones (un resultado común es cambiar de modo vertical al apaisado u horizontal).

iOS se deriva de Mac OS X, que a su vez está basado en Darwin BSD, y por lo tanto es un sistema operativo Unix.

iOS cuenta con cuatro capas de abstracción: la capa del núcleo del sistema operativo, la capa de "Servicios Principales", la capa de "Medios" y la capa de "Cocoa Touch". La versión actual del sistema operativo (iOS 6.0) ocupa más o menos 770 megabytes, variando por modelo.

Ilustración 3. Visualización de celular Apple Iphone

7.1.5.3. BlackBerry OS

Es un sistema operativo móvil desarrollado por Research In Motion para sus dispositivos BlackBerry. El sistema permite las multitareas y tiene soporte para

diferentes métodos de entrada adoptados por RIM para su uso en computadoras de mano, particularmente la *trackwheel*, *trackball*, touchpad y pantallas táctiles.

Su desarrollo se remonta la aparición de los primeros handheld en 1999. Estos dispositivos permiten el acceso a correo electrónico, navegación web y sincronización con programas como Microsoft Exchange o Lotus Notes aparte de poder hacer las funciones usuales de un teléfono móvil.

RIM estuvo en disputa con NTP Inc. la cual le acusaba de violar cinco patentes que pudo haber dejado sin servicio a sus usuarios en Estados Unidos (sobre tres millones). Las compañías llegaron a un acuerdo extrajudicial que soluciono la disputa en marzo de 2006 previo pago de 612 millones de dólares por parte de RIM. (Noticia)

Ilustración 4. Visualización de celular Blackberry

7.1.5.4. Symbian OS

Es un sistema operativo propietario diseñado para teléfonos móviles, con librerías asociadas e interface de usuario. Desciende de Psion EPOC y corre exclusivamente en procesadores ARM, es producido por Symbian Ltd, sociedad formada por Nokia (47.9%), Ericsson (15.6%), Sony Ericsson (13.1%), Panasonic (10.5%), Siemens AG (8.4%) y Samsung (4.5%).El objetivo de Symbian fue crear un sistema operativo para terminales móviles que pudiera competir con el de Palm o el Windows Mobile 6.X de Microsoft y ahora Android deGoogle Inc. , iOS de Apple Inc. y BlackBerry de RIM.

Ilustración 5. Visualización de celular con sistema operativo Symbian

Figura 7.

7.1.5.5. Microsoft Windows Mobile

Windows Mobile es un sistema operativo móvil compacto desarrollado por Microsoft, y diseñado para su uso en teléfonos inteligentes (*Smartphones*) y

otros dispositivos móviles. Intenta emular su interfaz al incluir un botón de inicio y una barra de tarea. Esto resulta poco práctico, ya que las pantallas reducidas pueden dificultar la visualización de los íconos tan pequeños.

El código de este sistema operativo es completamente cerrado. Sin embargo los desarrolladores pueden utilizar el API.net Mobile y el entorno de desarrollo visual estudio que es muy completo y permite el acceso a las funcionalidades de red de bajo nivel del dispositivo, aunque su licencia no es totalmente abierta.

Esta plataforma ha perdido la gran popularidad que disfrutaba hace algunos años debido a las malas políticas de mercadeo de Microsoft.[4]

Ilustración 6. Visualización de celular Windows Mobile

7.1.5.6. Linux

[4] http://www.javeriana.edu.co/biblos/tesis/ingenieria/tesis270.pdf (pág. 43)

Linux es un núcleo libre de sistema operativo basado en Unix Es uno de los principales ejemplos de software libre. Linux está licenciado bajo la GPL v2 y está desarrollado por colaboradores de todo el mundo. El desarrollo del día a día tiene lugar en la *Linux Kernel Mailing List Archive.*

El núcleo Linux fue concebido por el entonces estudiante de ciencias de la computación finlandés, Linus Torvalds, en 1991. Linux consiguió rápidamente desarrolladores y usuarios que adoptaron códigos de otros proyectos de software libre para su uso en nuevas distribuciones. El núcleo Linux ha recibido contribuciones de miles de programadores de todo el mundo. Normalmente Linux se utiliza junto a un empaquetado de software, llamado distribución Linux y servidores.

De acuerdo a la información anterior, entre las plataformas y tecnologías utilizadas en los dispositivos móviles mencionados anteriormente debe resaltarse que de todos los sistemas operativos que se mencionan a excepción de los iPhones OS, todas las plataformas facilitan la aplicación y el desarrollo de cualquier aplicación de forma gratuita y abierta, además que permiten el acceso a las funcionalidades de red de bajo nivel de los dispositivos.

Es claro que se debe tener en cuenta el lenguaje de programación para la implementación, por ejemplo en los RIM Blackberry OS, Symbian OS se desarrollan con tecnologia Java ME, pero Google Android OS se basa en Java SE.

Entonces para sigueindo la información antes explicada llegamos a la conclusión que la aplicación no se desarrollara para el sistema operativo Apple Phone OS, dado que las políticas de Apple no permite el desarrollo de ninguna aplicación compleja por parte de terceros.

La arquitectura que se diseñara e implementara para este proyecto utilizara Java ME y Java SE, ya que de esta forma quearàn cuebiertas las plataformas BlackBerry OS, Google Android OS y Symbian Os.[5]

- Android 1.6, creado por la *Open Handset Aliance* (liderada por Google), con licencia software Apache de software libre y código fuente abierto. Se caracteriza por estar desarrollado sobre un núcleo Linux. Aspecto más destacable: es una plataforma abierta que permite a cualquier fabricante desarrollar sus productos sobre ella.

- iPhone OS , creado por Apple, con licencia propietaria. Aspecto más destacable: Ha marcado tendencia en el mundo de la tecnología móvil y sigue destacando en él gracias a su interfaz de usuario y a su gran aprovechamiento de los recursos hardware del dispositivo sobre el que se ejecuta.

- Symbian OS, creado por Nokia, con licencia propietaria. A pesar de su aspecto renovado no deja de ser una nueva versión del clásico Symbian. Aspecto

[5] http://www.javeriana.edu.co/biblos/tesis/ingenieria/tesis270.pdf (pág. 45)

más destacable: a pesar de estar desarrollado por Nokia se puede encontrar en teléfonos de otros fabricantes.

- Blackberry OS , creado por RIM, con licencia propietaria. Se caracteriza por estar desarrollado en Java. Aspecto más destacable: es el sistema operativo líder en el sector empresarial, ya que está totalmente orientado al uso profesional.

- Windows Mobile, creado por Microsoft, con licencia propietaria. Una nueva versión del clásico sistema operativo de Windows para móviles. Su aspecto más destacable es el gran número de aplicaciones existentes tanto gratuitas como de pago.

7.1.6. Ecuaciones Lineales De Primer Grado

7.1.6.1. Qué Es Una Ecuación

En matemáticas, una ecuación es una igualdad[1] entre dos expresiones algebraicas, denominadas miembros, en las que aparecen valores conocidos o datos, y desconocidos o incógnitas, relacionados mediante operaciones matemáticas. Los valores conocidos pueden ser números, coeficientes o constantes; y también variables cuya magnitud se haya establecido como resultado de otras operaciones. Las incógnitas, representadas generalmente por letras, constituyen los valores que se pretende hallar.

La variable representa la incógnita, mientras que el coeficiente y los números son constantes conocidas. La igualdad planteada por una ecuación será cierta o

falsa dependiendo de los valores numéricos que tomen ambos miembros; se puede afirmar entonces que una ecuación es una igualdad condicional, en la que solo ciertos valores de las variables la hacen cierta.

Se llama solución de una ecuación a cualquier valor individual de dichas variables que la satisfaga.

Resolver una ecuación es encontrar su dominio solución, que es el conjunto de valores de las incógnitas para los cuales la igualdad se cumple. Todo problema matemático puede expresarse en forma de una o más ecuaciones; sin embargo no todas las ecuaciones tienen solución, ya que es posible que no exista ningún valor de la incógnita que haga cierta una igualdad dada. En ese caso, el conjunto de soluciones de la ecuación será vacío y se dice que la ecuación no es resoluble. De igual modo, puede tener un único valor, o varios, o incluso infinitos valores, siendo cada uno de ellos una solución particular de la ecuación. Si cualquier valor de la incógnita hace cumplir la igualdad (esto es, no existe ningún valor para el cual no se cumpla) la expresión se llama identidad.[2]

La ciencia utiliza ecuaciones para enunciar de forma precisa leyes; estas ecuaciones expresan relaciones entre variables. Así, en física, la ecuación de la dinámica de Newton relaciona las variables fuerza F, aceleración a y masa m: F = ma. Los valores que son solución de la ecuación anterior cumplen al primera ley de la mecánica de Newton. Por ejemplo, si se considera una masa m = 1 kg y una

aceleración a = 1 m/s, la única solución de la ecuación es F = 1 Kg·m/s = 1 Newton, que es el único valor para la fuerza permitida por la ley.

El campo de aplicación de las ecuaciones es inmenso, y por ello hay una gran cantidad de investigadores dedicados a su estudio.[6]

7.1.6.2. Tipos de ecuaciones

Las ecuaciones pueden clasificarse según el tipo de operaciones necesarias para definirlas y según el conjunto de números sobre el que se busca la solución. Entre los tipos más frecuentes están:[7]

- Ecuaciones Algebraicas
- Polinómicas o polinomiales
- Primer grado o lineales
- Segundo grado o cuadráticas Racionales, aquellas en las que uno o ambos miembros se expresan como un cociente de polinomios.
- Ecuaciones trascendentes, cuando involucran funciones no polinómicas, como las trigonométricas, exponenciales, etc,
- Diofánticas o diofantinas
- Ecuaciones diferenciales
- Ordinarias
- Ecuaciones derivadas parciales
- Ecuaciones integrales

[6] http://celutron.blogspot.com/2007/11/manos-la-obra-anatoma-de-una-aplicacin.html

[7] http://www.youtube.com/watch?v=wwlHv_9yajo&feature=related

7.1.6.3. Ecuaciones de primer grado

Una ecuación es una igualdad que sólo se verifica para unos valores concretos una variable, generalmente llamada x.

Resolver una ecuación consiste en hallar los valores de la variable que hacen cierta la igualdad. Recuerda: Si un elemento está sumando en un miembro pasa al otro restando. Si está restando pasa sumado. Si un número multiplica a todos los elementos de un miembro pasa al otro dividiendo y si los divise pasa multiplicando.[8]

7.1.6.4. Formas de ecuaciones lineales

Formas complejas como las anteriores pueden reescribirse usando las reglas del álgebra elemental en formas más simples. Las letras mayúsculas representan constantes, mientras x e y son variables.

Aquí A y B no son ambos cero. Representa una línea en el cartesiano. Es posible encontrar los valores donde x e y se anulan.

Ecuación segmentaria o simétrica, aquí ni E ni F no pueden ser cero. El gráfico de esta ecuación corta al eje X y al eje Y en E y F respectivamente.

Forma paramétrica las ecuaciones que deben cumplirse de manera simultánea, cada una en la variable t. Puede convertirse a la forma general despejando t en ambas ecuaciones e igualando.

[8] http://www.youtube.com/watch?v=n2ebqjrckjw&feature=rec-LGOUT-exp_fresh+div-1r-3-HM

7.1.6.5. Casos especiales

Un caso especial es la forma estándar donde Y. El gráfico es una línea horizontal sin intersección con el eje X ó (si F = 0) coincidente con ese eje.

Otro caso especial de la forma general donde Y. El gráfico es una línea vertical, interceptando el eje X en E.

En este caso, todas las variables fueron canceladas, dejando una ecuación que es verdadera en todos los casos. La forma original (no una tan trivial como la del ejemplo), es llamada identidad. El gráfico es todo el plano cartesiano, ya que lo satisface todo par de números reales x e y.

Nótese que si la manipulación algebraica lleva a una ecuación como 1 = 0 entonces la original es llamada inconsistente, o sea que no se cumple para ningún par de números x e y. Adicionalmente podría haber más de dos variables, en ecuaciones simultaneas Ecuación lineal en el espacio n-dimensional son funciones lineales de varias variables que admiten también interpretaciones geométricas.

7.1.6.6. Sistemas de ecuaciones lineales

Los sistemas de ecuaciones lineales expresan varias ecuaciones lineales simultáneamente y admiten un tratamiento matricial. Para su resolución debe haber tantas ecuaciones como incógnitas y el determinante de la matriz ha de ser real y no nulo. Geométricamente corresponden a intersecciones de líneas en un

único punto (sistema lineal de dos ecuaciones con dos incógnitas), planos en una recta (dos ecuaciones lineales de tres incógnitas) o un único punto (tres ecuaciones lineales de tres incógnitas). Los casos en los que el determinante de la matriz es nulo no poseen solución.[9]

7.1.7. Pendiente

7.1.7.1. Qué es una pendiente

Pendiente: En matemáticas y ciencias aplicadas se denomina pendiente a la inclinación de un elemento ideal, natural o constructivo respecto de la horizontal.

En geometría, puede referirse a la *pendiente de la ecuación de una recta* como caso particular de la tangente a una curva, en cuyo caso representa la derivada de la función en el punto considerado, y es un parámetro relevante, por ejemplo, en el trazado altimétrico de carreteras, vías férreas o canales.

7.1.7.2. Pendiente de una recta

La pendiente de una recta en un sistema de representación rectangular (de un plano cartesiano), suele ser representado por la letra m, y es definido como el cambio o diferencia en el eje Y dividido por el respectivo cambio en el eje X, entre 2 puntos de la recta. En la siguiente ecuación se describe: toda recta que no sea horizontal, tiene que cortar al eje "x". Se dice que si una recta corta al eje X, la inclinación de la recta se define como el ángulo positivo menor de 180°.

$$m = \frac{\Delta y}{\Delta x}$$

[9] Weisstein, Eric W. «Ecuación lineal» (en inglés). MathWorld. Wolfram Research.

7.1.7.3. Geometría

Una recta horizontal tiene pendiente igual a 0 (cero). Cuanto menor sea el valor de la pendiente, menor inclinación tendrá la recta; por ejemplo, una recta que se eleve un ángulo de 45° con respecto al eje X tiene una pendiente m = +1, y una recta que caiga 30° tiene pendiente m = -0,5. La pendiente de una recta vertical no está definida, o se dice que es infinita.

El ángulo θ que una recta forma con el eje horizontal está relacionado con la pendiente *m* por medio de la siguiente relación trigonométrica:

$$m = \tan \theta$$

o equivalentemente:

$$\theta = \arctan m$$

Dos o más rectas son paralelas si ambas poseen la misma pendiente, o si ambas son verticales y por ende no tienen pendiente definida; dos o más rectas son perpendiculares (forman un ángulo recto entre ellas) si el producto de sus pendientes es igual a -1.

7.1.7.4. La pendiente en las ecuaciones de la recta

Si *y* es una función lineal de *x*, entonces el coeficiente de *x* es la pendiente de la recta. Por lo tanto, si la ecuación está dada de la siguiente manera:

$$y = mx + b$$

Entonces *m* es la pendiente. En esta ecuación, el valor de b puede ser interpretado como el punto donde la recta se intersecta con el eje Y, es decir, el valor de y cuándo$x = 0$. Este valor también es llamado coordenada de origen.

Si la pendiente m de una recta y el punto (x_0, y_0) de la recta son conocidos, entonces la ecuación de la recta puede ser encontrada usando:

$$y - y_0 = m(x - x_0)$$

La pendiente de la recta en la fórmula general:

$$Ax + By + C = 0$$

Está dada por:

$$m = -\frac{A}{B}$$

7.1.8. Arquitecturas De Software

Una definición reconocida es la de Clements [Cle96a]: La arquitectura de software es, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones.

La Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema, programa o aplicación y tiene la responsabilidad de: definir los módulos principales, definir las responsabilidades que tendrá cada uno de estos módulos, definir la interacción que existirá entre dichos módulos: Control y flujo de datos, Secuenciación de la información, Protocolos de interacción y comunicación,

Ubicación en el hardware. La Arquitectura del Software aporta una visión abstracta de alto nivel, posponiendo el detalle de cada uno de los módulos definidos a pasos posteriores del diseño.

La definición oficial de Arquitectura del Software es la IEEE Std 1471-2000 dice: “La Arquitectura del Software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución”.

7.1.8.1. Tipos de Arquitectura

Cada arquitectura conocida puede ser utilizada en función de sus ventajas e inconvenientes para cada caso en concreto. Así, las arquitecturas más universales son:

Monolítica. Donde el software se estructura en grupos funcionales muy acoplados.

Cliente-servidor. Donde el software reparte su carga de cómputo en dos partes independientes pero sin reparto claro de funciones.

Arquitectura de tres niveles. Especialización de la arquitectura cliente-servidor donde la carga se divide en tres partes (o capas) con un reparto claro de funciones: una capa para la presentación (interfaz de usuario), otra para el cálculo (donde se encuentra modelado el negocio) y otra para el almacenamiento (persistencia). Una capa solamente tiene relación con la siguiente.

Otras arquitecturas conocidas son:

- En pipeline
- Entre pares
- En pizarra
- Orientada a servicios
- Máquinas virtuales

7.1.8.2. Características de las Arquitecturas

- La Arquitectura de Software establece los fundamentos para que analistas, diseñadores, programadores, etc. trabajen en una línea común que permita alcanzar los objetivos del sistema de información, cubriendo todas las necesidades.
- Una arquitectura de software se selecciona y diseña con base en objetivos y restricciones.
- Los objetivos son prefijados para el sistema de información, pero no solamente los de tipo funcional, también otros objetivos como la mantenibilidad, audibilidad, flexibilidad e interacción con otros sistemas de información.
- Las restricciones son limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información.
- La arquitectura de software define, de manera abstracta, los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación ente ellos.

- Toda arquitectura debe ser implementable en una arquitectura física, que consiste simplemente en determinar qué computadora tendrá asignada cada tarea.

7.1.9. Descripción De Arquitecturas

7.1.9.1. Orientada a Servicios

Algunas definiciones de esta arquitectura:

W3C: “Conjunto de componentes que pueden ser invocados, cuyas descripciones de interfaces se pueden publicar y descubrir”

CBDI rechaza esa definición: Los componentes pueden no ser conjuntos. La definición sólo considera los componentes y no la práctica o el arte de construir la arquitectura. Para CBDI: “Estilo resultante de políticas, prácticas y frameworks que permiten que la funcionalidad de una aplicación se pueda proveer y consumir como conjuntos de servicios, con una granularidad relevante para el consumidor. Los servicios pueden invocarse, publicarse y descubrirse y están abstraídos de su implementación utilizando una sola forma estándar de interface”.

“Infraestructura de alto nivel basada en best practices y patrones para crear soluciones basadas en servicios, de alta cohesión y bajo acoplamiento” (Geniant®). “Estilo arquitectónico apto para implementar bajo acoplamiento entre agentes. Los agentes son proveedores y consumidores de servicios, que son la unidad de trabajo”. (Hao He). “Una arquitectura de aplicación en la cual todas las

funciones se definen como servicios independientes con interfaces invocables bien definidas, que pueden ser llamadas en secuencias definidas para formar procesos de negocios" (IBM).

MITRE: Una aplicación SOA es una colección de servicios. Un servicio es la unidad atómica de una SOA Los servicios encapsulan procesos de negocios. Los proveedores de servicios se registran solos. Un servicio involucra: Find, Bind, Execute. Las instancias más conocidas son los web services.

Gartner: "SOA es una arquitectura de software que comienza con una definición de interface y construye toda la topología de la aplicación como una topología de interfaces, implementaciones y llamados a interfaces. Sería mejor llamada "arquitectura orientada a interfaces". SOA es una relación de servicios y consumidores de servicios, ambos suficientemente amplios para representar una función de negocios completa".

Estructura

Las Arquitecturas Orientadas a Servicios, están motivadas por la creciente necesidad de los negocios de responder con rapidez a los cambios en el entorno comercial en que se desenvuelven. Esto los lleva a tener que cambiar sus sistemas tecnológicos con esa misma rapidez y para lograrlo es necesario que los

componentes de esta infraestructura, sean tan reutilizables y poco interdependientes que permitan una rápida reestructuración de los mismos.

Los elementos básicos que conforman SOA son:

- Proveedores de Servicios
- Consumidores de Servicios
- Bus Empresarial de Servicios

Además podemos definir otros elementos participantes dentro de una arquitectura SOA tales como:

- Cliente – proveedor Se entiende Cliente como el componente que invoca un servicio provisto por un proveedor.
- Concepto "Especialización de componente de software". Especializar componentes para que ejecuten tareas identificadas y específicas.
- Concepto de servicio Un servicio es una unidad de trabajo realizada por un componente de software a fi n de conseguir un resultado específico. El servicio debe ser alcanzable por parte de los consumidores a través de una interfaz programática.
- Utilización de webservices La arquitectura orientada a servicios, no especifica necesariamente que los servicios deben ser brindados a través de un protocolo específico. Los Web Services son en realidad un conjunto de estándares que definen un protocolo de invocación remota de servicios, basados en HTML y XML. Si bien, son también un mecanismo adecuado y en muchos casos

recomendable para implementar servicios no son el único. Es importante que las arquitecturas orientadas a servicios soporten múltiples protocolos a fin de cumplir al máximo su visión de brindar un modelo de integración para toda la plataforma tecnológica.

7.1.9.2. Alcance de una SOA

SOA no define en si mismo, todo lo que una organización debe contemplar en su ambiente tecnológico. Otras mejores prácticas, como una buena metodología de ingeniería de software, por ejemplo RUP (Racional Unified Process) o ITIL, deben acompañar a cualquier iniciativa de implementar SOA en una organización.

Cada vez más las empresas se ven obligadas a ser organizaciones On Demand... "Una empresa on demand para IBM, es una organización cuyos procesos de negocio-integrados de principio a fin a través de la compañía y con aliados clave, proveedores y clientes- pueden responder con velocidad a cualquier cambio en las demandas de los clientes, oportunidades de mercado o amenazas externas". La necesidad de tener esta rápida capacidad de respuesta, fundamentada en los procesos del negocio es cada vez más el diferenciador clave del éxito empresarial.

Alcance a nivel de proceso de negocios ofrece esta arquitectura

Tal y como se evidencia de la definición de una empresa On Demand, la capacidad de respuesta está fundamentada en los procesos del negocio. Es por esto que IBM, así como la mayor parte de la industria, ha venido a relacionar SOA con conceptos como Administración de Procesos del Negocio y Desempeño del

Negocio (Business Process Management y Business Performance Management). El valor que esto brinda a las organizaciones que adoptan SOA, es la posibilidad de ver su organización como un conjunto de procesos que involucran los diversos servicios, componentes, recursos o personas a través de toda la empresa, y a partir de esto ser capaces de medir su desempeño en cada paso.

Se puede decir que los 2 elementos que brindan el mayor valor agregado de esta arquitectura: el débil acoplamiento entre los servicios (componentes), que redundan en una mayor velocidad de respuesta, (facilidad para planear y ejecutar cambios) y en el aumento en la reutilización de los recursos tecnológicos, que implica una reducción en los costos y un mejor retorno de las inversiones en tecnología.

SOA brinda el marco conceptual sobre el que se pueden gestar nuevos modelos tecnológicos que respondan a las crecientes necesidades tecnológicas del mundo empresarial actual. Sobre este modelo están evolucionando los nuevos modelos en tecnologías de información y sistemas empresariales.

Esto concuerda con un estudio realizado entre diciembre del 2005 y enero de 2006, por IDG Research, entre más de mil usuarios de tecnologías en Estados Unidos. Un 52% de ellos afirmó que SOA es una prioridad "crítica" o "alta" en sus planes para los próximos doce meses. Incluso el sondeo detectó que en los últimos nueve meses, la cantidad de despliegues de alcance corporativo de SOA se han duplicado, pasando de un 8% a un 16%, mientras que las compañías que afirman haber creado un grupo de "arquitectura corporativa" pasaron de un 68% a un 83%.

7.1.9.3. Cliente/Servidor

La arquitectura cliente-servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, que le da respuesta. Esta idea también se puede aplicar a programas que se ejecutan sobre una sola computadora, aunque es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.[10]

FUNCIONAMIENTO

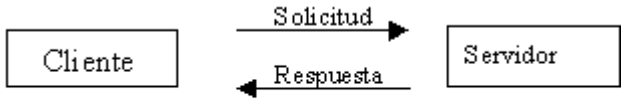
La tecnología cliente/servidor es el procesamiento cooperativo de la información por medio de un conjunto de procesadores, en el cual múltiples clientes, distribuidos geográficamente, solicitan requerimientos a uno o más servidores centrales. Desde el punto de vista funcional, se puede definir la computación cliente/servidor como una arquitectura distribuida que permite a los usuarios

[10] http://es.wikipedia.org/wiki/Cliente-servidor

finales obtener acceso a la información en forma transparente aun en entornos multiplataforma.

En el modelo cliente servidor, el cliente envía un mensaje solicitando un determinado servicio a un servidor, y este envía uno o varios mensajes con la respuesta. En un sistema distribuido cada máquina puede cumplir el rol de servidor para algunas tareas y el rol de cliente para otras. Además como veremos en el modelo de implementación, el concepto es utilizado en forma constante para varias funciones e implementado de distintas formas.

Ilustración 7. Modelo Cliente - servidor



Modelo cliente servidor

La idea es tratar a una computadora como un instrumento, que por sí sola pueda realizar muchas tareas, pero con la consideración de que realice aquellas que son más adecuadas a sus características. Si esto se aplica tanto a clientes como servidores se entiende que la forma más estándar de aplicación y uso de sistemas clientes/servidores es mediante la explotación de las PC a través de interfaces gráficas de usuario; mientras que la administración de datos y su seguridad e integridad se deja a cargo de computadoras centrales tipo mainframe.

Como se desprende de las definiciones anteriores, tanto clientes como servidores son entidades independientes que operan conjuntamente a través de una red para realizar una tarea. Pero para hacer la distinción respecto de otras formas de arquitecturas o software distribuidos, se presenta una lista de características que debieran cumplir los sistemas cliente/servidor:

- Se establece una relación entre procesos distintos, los cuales pueden ser ejecutados en la misma máquina o en máquinas diferentes distribuidas a lo largo de la red.
- Existe una clara distinción de funciones basada en el concepto de "servicio", que se establece entre clientes y servidores.
- La relación establecida puede ser de muchos a uno, en la que un servidor puede dar servicio a muchos clientes, regulando su acceso a recursos compartidos.
- Los clientes corresponden a procesos activos en cuanto a que son éstos lo que hacen peticiones de servicios a los servidores.
- Estos últimos tienen un carácter pasivo ya que esperan las peticiones de los clientes. No existe otra relación entre clientes y servidores que no sea la que se establece a través del intercambio de mensajes entre ambos.
- El mensaje es el mecanismo para la petición y entrega de solicitudes de servicio. Las plataformas de software y hardware entre clientes y servidores son independientes.

- Precisamente una de las principales ventajas de esta arquitectura es la posibilidad de conectar clientes y servidores independientemente de sus plataformas.
- El concepto de escalabilidad tanto horizontal como vertical es aplicable a cualquier sistema cliente/servidor.
- La escalabilidad horizontal permite agregar más estaciones de trabajo activas sin afectar significativamente el rendimiento. La escalabilidad vertical permite mejorar las características del servidor o agregar múltiples servidores.

CARACTERISTICAS

- Flexibilidad

Middleware.

Separación de funciones:

- Lógica de presentación
- Lógica de negocio
- Lógica de datos

Encapsulación de servicios

Portabilidad - reubicación

Operación síncrono - asíncrono

- Entorno de aplicaciones incremental

Añadir un nuevo servidor

Añadir un nuevo cliente

Modificar un cliente para usar un nuevo servidor

Integración: por la GUI

7.1.9.4. MIDDLEWARE

En su definición más simple, middleware es la interfaz que provee la conectividad entre aplicaciones clientes y aplicaciones servidoras, y entre aplicaciones y bases de datos. Es una capa de software que protege a los desarrolladores de tener que manejar detalles de bajo nivel de diferentes protocolos de comunicación, sistemas operativos y arquitecturas de bases de datos. Este tipo de interfaces incluyen API's, PRC's, Pipes, mensajería de red y accesos a bases de datos.

- Conecta procesos para constituir aplicación
- Conjunto de funciones + servicios
- Actúa en el bajo nivel del SID:

Comunicación.

Directorios.

Integridad.

- Define la plataforma de transparencia de localización.

NIVELES

Para entender en forma más ordenada y clara los conceptos y elementos involucrados en esta tecnología se puede aplicar una descomposición o arquitectura de niveles. Esta descomposición principalmente consiste en separar los elementos estructurales de esta tecnología en función de aspectos más funcionales de la misma:

- Nivel de Presentación: Agrupa a todos los elementos asociados al componente Cliente.
- Nivel de Aplicación: Agrupa a todos los elementos asociados al componente Servidor.
- Nivel de comunicación: Agrupa a todos los elementos que hacen posible la comunicación entre los componentes Cliente y servidor.
- Nivel de base de datos: Agrupa a todas las actividades asociadas al acceso de los datos.
- Este modelo de descomposición en niveles, como se verá más adelante, permite introducir más claramente la discusión del desarrollo de aplicaciones en arquitecturas de hardware y software en planos.

Cliente

El cliente es el proceso que permite al usuario formular los requerimientos y pasarlos al servidor, se lo conoce con el término front-end. Este normalmente maneja todas las funciones relacionadas con la manipulación y despliegue de datos, por lo que están desarrollados sobre plataformas que permiten construir interfaces gráficas de usuario (GUI), además de acceder a los servicios distribuidos en cualquier parte de la red. Las funciones que lleva a cabo el proceso cliente se resumen en los siguientes puntos:

- Administrar la interfaz de usuario.
- Interactuar con el usuario.
- Procesar la lógica de la aplicación y hacer validaciones locales.
- Generar requerimientos de bases de datos. Recibir resultados del servidor.

- Formatear resultados

Servidor

Es el proceso encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. Al proceso servidor se lo conoce con el término back-end. El servidor normalmente maneja todas las funciones relacionadas con la mayoría de las reglas del negocio y los recursos de datos.

Las funciones que lleva a cabo el proceso servidor se resumen en los siguientes puntos:

- Aceptar los requerimientos de bases de datos que hacen los clientes.
- Procesar requerimientos de bases de datos.
- Formatear datos para trasmitirlos a los clientes.
- Procesar la lógica de la aplicación y realizar validaciones a nivel de bases de datos.

Clasificación de modelos Cliente/Servidor

Uno de los aspectos claves para entender la tecnología cliente/servidor, y por lo tanto contar con la capacidad de proponer, promocionar y llevar a cabo soluciones de este tipo, es llegar a conocer la arquitectura de este modelo y los conceptos o ideas asociados al mismo. Más allá de entender los componentes cliente/middleware/servidor, es preciso analizar ciertas relaciones entre éstos, que pueden definir el tipo de solución que se ajusta de mejor forma a las estadísticas y

restricciones acerca de los eventos y requerimientos de información que se obtuvieron en la etapa de análisis de un determinado proyecto. De hecho el analista o líder deberá conocer estos eventos/restricciones del negocio para, a partir de allí, hacer las consideraciones y estimaciones de la futura configuración, teniendo en cuenta aspectos como por ejemplo, la oportunidad de la información, tiempo de respuesta, tamaños de registros, tamaño de bases de datos, estimaciones del tráfico de red, distribución geográfica tanto de los procesos como los datos, etc.

En tal sentido se presenta, en primer lugar, un esquema de clasificación basado en los conceptos de Fat Client/Thin Client, Fat Server/Thin Server, Two Tier, Three Tier, los cuales están bastante generalizados y sobrecargados de definiciones, pero que se consideran necesarios y útiles para la aplicación del modelo cliente/servidor. Después se presentará un esquema de clasificación según otros aspectos.

Por tamaño de componentes

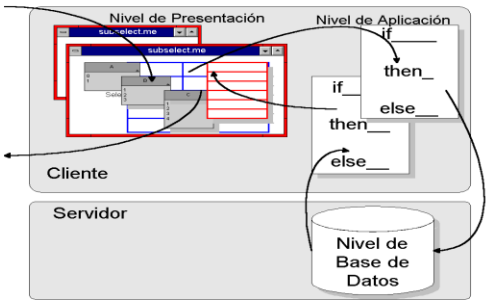
Este tipo de clasificación se basa en los grados de libertad que brinda el modelo cliente/servidor para balancear la carga de proceso entre los niveles de presentación, aplicación y base de datos. Dependiendo de qué segmento de las capas de software tenga que soportar la mayor o menor carga de procesamiento, se habla de Fat Cliente (Thin Server) o Fat server (Thin Client). Consideraciones de este tipo son importantes al momento de decidir una plataforma de

desarrollo/explotación, al punto que pueden definir la viabilidad o no de las mismas para enfrentar un cierto número de restricciones impuestas por una problemática a resolver.

Fat Client (Thin Server)

En este esquema de arquitectura el grueso de la aplicación es ejecutada en el cliente, es decir, el nivel de presentación y el nivel de aplicación corren en un único proceso cliente, y el servidor es relegado a realizar las funciones que provee un administrador de base de datos.

Ilustración 8. Modelo de nivel de aplicación y presentación de cliente - servidor

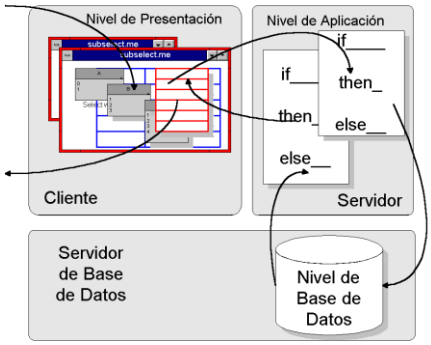


En general este tipo de arquitectura tiene mejor aplicación en sistemas de apoyo de decisiones (DSS: Decision Support System) y sistemas de información ejecutiva (EIS: Executive Information System), y como se concluirá más adelante, tiene pocas posibilidades de aplicarse en sistemas de misión crítica.

Fat Server (Thin Client)

Este es el caso opuesto al anterior, el proceso cliente es restringido a la presentación de la interfaz de usuario, mientras que el grueso de la aplicación corre por el lado del servidor de aplicación.

Ilustración 9. Proceso de cliente restringido y aplicación corre por el servidor



En general este tipo de arquitectura presenta una flexibilidad mayor como para desarrollar un gran espectro de aplicaciones, incluyendo los sistemas de misión crítica a través de servidores de transacciones.

Por planos o capas (Tier)

Una de las más comunes y discutidas distinciones entre las diferentes arquitecturas cliente/servidor se basan en la idea de planos (tier), la cual es una variación sobre la división o clasificación por tamaño de componentes (clientes grandes y servidores amplios). Esto se debe a que se trata de definir el modo en que las prestaciones funcionales de la aplicación serán asignadas, y en qué proporción, tanto al cliente como al servidor. Dichas prestaciones se deben agrupar entre los tres componentes clásicos para cliente/servidor: interfaz de

usuario, lógica de negocios y los datos compartidos, cada uno de los cuales corresponde a un plano.

Dentro de esta categoría tenemos las aplicaciones en dos planos (two-tier), tres planos (three-tier) y multi planos (multi-tier). Dado que este término ha sido sobrecargado de significados por cuanto se lo utiliza indistintamente para referirse tanto a aspectos lógicos (Software) como físicos (Hardware), aquí se esquematizan ambas acepciones.

Planos a niveles de software

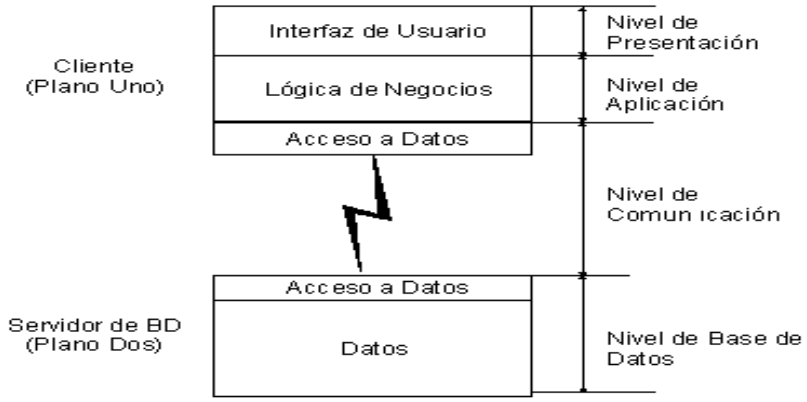
Este enfoque o clasificación es el más generalizado y el que más se ajusta a los enfoques modernos, dado que se fundamenta en los componentes lógicos de la estructura cliente/servidor y en la madurez y popularidad de la computación distribuida. Por ejemplo, esto permite hablar de servidores de aplicación distribuidos a lo largo de una red, y no tiene mucho sentido identificar a un equipo de hardware como servidor, si no más bien entenderlo como una plataforma física sobre la cual pueden operar uno o más servidores de aplicaciones.

7.1.9.5. Cliente/Servidor Dos Planos

Esta estructura se caracteriza por la conexión directa entre el proceso cliente y un administrador de bases de datos. Dependiendo de dónde se localice el grupo de

tareas correspondientes a la lógica de negocios se pueden tener a su vez dos tipos distintos dentro de esta misma categoría.

Ilustración 10. Implementado con SQL Remoto



En este esquema el cliente envía mensajes con solicitudes SQL al servidor de bases de datos y el resultado de cada instrucción SQL es devuelto por la red, no importando si son uno, diez, cien o mil registros. Es el mismo cliente quien debe procesar todos los registros que le fueron devueltos por el servidor de base de datos, según el requerimiento que él mismo hizo. Esto hace que este tipo de estructura se adecue a los requerimientos de aplicaciones orientadas a los sistemas de apoyo y gestión, pero resultan inadecuados para los sistemas críticos en que se requieran bajos tiempos de respuesta.

Ventajas:

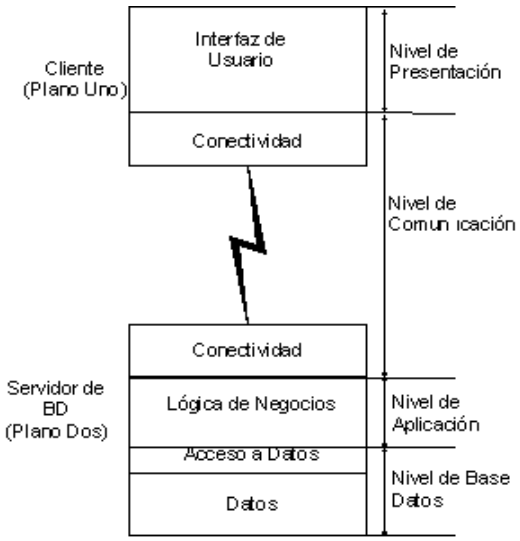
- Presenta una estructura de desarrollo bastante simple por cuanto el programador típicamente maneja un único ambiente de desarrollo (es más simple

respecto de cliente/servidor en tres planos, puesto que reduce una capa de programación, como se verá más adelante).

Desventajas:

- La gran cantidad de información que viaja al cliente congestiona demasiado el tráfico de red, lo que se traduce en bajo rendimiento.
- Por su bajo rendimiento esta estructura tiene un bajo espectro de aplicación, limitándose a la construcción de sistemas no críticos.

Ilustración 11. Implementado con Procedimientos Almacenados



En este esquema el cliente envía llamadas a funciones que residen en la base de datos, y es ésta quien resuelve y procesa la totalidad de las instrucciones SQL agrupadas en la mencionada función.

Ventajas:

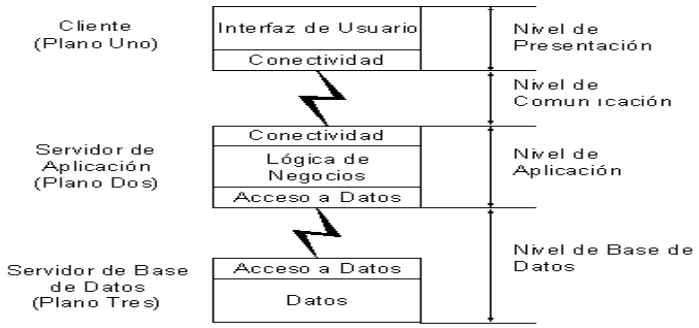
- Presenta las mismas ventajas de una arquitectura dos planos con procedimientos almacenados, pero mejora considerablemente el rendimiento sobre ésta, dado que reduce el tráfico por la red al procesar los datos en la misma base de datos, haciendo viajar sólo el resultado final de un conjunto de instrucciones SQL.

Desventajas:

- Si bien la complejidad de desarrollo se ve disminuida, se pierde flexibilidad y escalabilidad en las soluciones implantadas (especialmente respecto de cliente/servidor en tres planos, como se verá mas adelante).
- Obliga a basar el grueso de la aplicación en SQL extendido, propios del proveedor de la base de datos que se elija. Debiera considerarse que sí bien los procedimientos almacenados (stored procedures), los desencadenantes (triggers) y las reglas (constraint) son útiles, en rigor son ajenos al estándar de SQL:
- No existen dos implementaciones de proveedores iguales.
- El lenguaje para la descripción de los procedimientos almacenados y probablemente su funcionalidad varía de un proveedor a otro. Lo que implica que los procedimientos almacenados no son totalmente exportables entre plataformas de distintos proveedores.
- Se pierde la independencia entre el código de la aplicación (conocimiento y reglas del negocio) y los datos.

7.1.9.6. Cliente/Servidor Tres Planos.

Ilustración 12. Modelo cliente - servidor de 3 planos



Esta estructura se caracteriza por elaborar la aplicación en base a dos capas principales de software, más la capa correspondiente al servidor de base de datos. Al igual que en la arquitectura dos capas, y según las decisiones de diseño que se tomen, se puede balancear la carga de trabajo entre el proceso cliente y el nuevo proceso correspondiente al servidor de aplicación.

En el esquema el cliente envía mensajes directamente al servidor de aplicación el cual debe administrar y responder todas las solicitudes. Es el servidor, dependiendo del tipo de solicitud, quien accede y se conecta con la base de datos.

Ventajas:

- Reduce el tráfico de información en la red por lo que mejora el rendimiento de los sistemas (especialmente respecto de la estructura en dos planos).

- Brinda una mayor flexibilidad de desarrollo y de elección de plataformas sobre la cual montar las aplicaciones.
- Provee escalabilidad horizontal y vertical.
- Se mantiene la independencia entre el código de la aplicación (reglas y conocimiento del negocio) y los datos, mejorando la portabilidad de las aplicaciones.
- Los lenguajes sobre los cuales se desarrollan las aplicaciones son estándares lo que hace más exportables las aplicaciones entre plataformas.
- Dado que mejora el rendimiento al optimizar el flujo de información entre componentes, permite construir sistemas críticos de alta confiabilidad.
- El mismo hecho de localizar las reglas del negocio en su propio ambiente, en vez de distribuirlos en la capa de interfaz de usuario, permite reducir el impacto de hacer mantenimiento, cambios urgentes de última hora o mejoras al sistema.
- Disminuye el número de usuarios (licencias) conectados a la base de datos.

Desventajas:

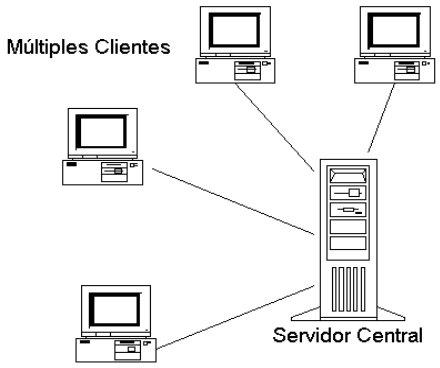
- Dependiendo de la elección de los lenguajes de desarrollo, puede presentar mayor complejidad en comparación con cliente/servidor dos planos.
- Existen pocos proveedores de herramientas integradas de desarrollo con relación al modelo cliente/servidor dos planos, y normalmente son de alto costo.
- Debido a estas desventajas es aquí la mayor importancia del Generador (aplicación creada en el presente trabajo).

Planos a niveles de hardware

Esta clasificación del modelo cliente/servidor se basa igualmente en la distribución de los procesos y elementos entre sus componentes, pero centrándose en la parte física del mismo, en el que la administración de la interfaz gráfica se asocia a los clientes PC y la seguridad e integridad de los datos quedan asociados a ambientes mainframe o por lo menos a servidores locales y/o centrales.

7.1.9.7. Cliente/Servidor Dos Planos

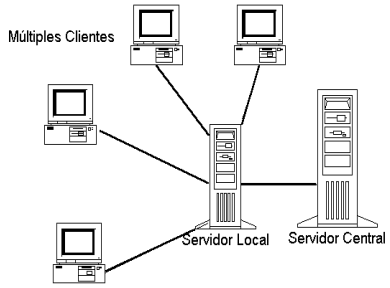
Ilustración 13. Modelo cliente - servidor dos planos



Como se ve en la figura, los clientes son conectados vía LAN a un servidor de aplicaciones local, el cual, dependiendo de la aplicación puede dar acceso a los datos administrados por él.

7.1.9.8. Cliente/Servidor Tres Planos

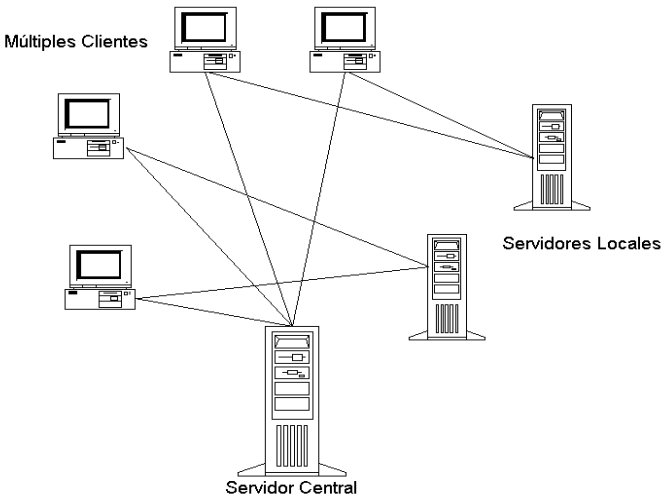
Ilustración 14. Arquitectura de cliente - servidor en tres planos



Como se ve en la figura, los clientes son conectados vía LAN a un servidor de aplicaciones local, el cual a su vez se comunica con un servidor central de bases de datos. El servidor local tiene un comportamiento dual, dado que actúa como cliente o servidor en función de la dirección de la comunicación.

7.1.9.9. Cliente/Servidor Múltiples Planos

Ilustración 15. Modelo de cliente - servidor en multiples planos



Este esquema permite que las PCs clientes puedan conectarse directamente a un servidor de bases de datos, pasando por alto a los servidores locales, los cuales son utilizados como simples servidores de archivos.

Cabe aclarar que este enfoque no se contrapone con el enfoque centrado en planos lógicos, es decir, la clasificación del modelo cliente/servidor en planos a niveles de software, sino más bien son distintos puntos de vista que se complementan al momento de diseñar soluciones cliente/servidor. En tal sentido, debe entenderse que la clasificación de la arquitectura cliente/servidor en "planos a niveles de hardware", define la configuración física más conveniente sobre la cual montar una determinada aplicación, la cual es estructurada de acuerdo a las consideraciones del modelo lógico o "planos a niveles de software".

Por la naturaleza del servicio

Servidores de Bases de Datos

Este análisis está elaborado desde el punto de vista del modelo cliente/servidor, y está directamente relacionado con la arquitectura en dos planos, descrita anteriormente. Dado que las características inherentes a dicho modelo ya fueron dadas, aquí sólo se hará una extensión a otros aspectos que se consideran relevantes.

Obviamente la creación de aplicaciones cliente/servidor está asociada a la utilización de servidores de bases de datos relacionales SQL, y dependiendo de

los requerimientos y restricciones se debe elegir entre una arquitectura dos o tres planos. Pero una arquitectura centrada en un servidor de bases de datos, cualquiera de las modalidades dos planos, permite que un proceso cliente solicite datos y servicios directamente a un servidor de bases de datos. Según las distintas normas de SQL (SQL-89, SQL-92, SQL3) el servidor debe proveer un acceso compartido a los datos con los mecanismos de protección necesarios, así como proveer mecanismos para seleccionar resultados dentro de un conjunto de datos, posibilitando un ahorro en procesos de comunicación. El servidor debe también proveer mecanismos de concurrencia, seguridad y consistencia de datos, basado principalmente en el concepto de transacción en el que todo se realiza, y por lo tanto se hace permanente, o todo falla, anulándose la transacción en tal caso.

Los servidores de bases de datos actuales son una mezcla de SQL estándar más otras extensiones propias de cada proveedor. Por ejemplo casi todas las bases de datos están provistas con procedimientos almacenados (stored procedures), desencadenantes (triggers) y restricciones (constraints) pero presentan diferencias importantes en su implementación. Es claro que esto obedece a presiones comerciales para tratar de extender los mecanismos de bases de datos para que realicen más funciones de las que corresponden a un servidor SQL relacional, con el objeto de tener una mayor participación en el espectro de los sistemas cliente/servidor por parte de los proveedores de bases de datos y como una manera de diferenciar sus productos.

Procedimientos Almacenados

Una de las posibilidades de implementar de mejor forma un sistema cliente/servidor en dos planos (two-tier), sin olvidar todas sus restricciones y limitaciones, es a través de procedimientos almacenados, que son funciones que agrupan un conjunto de instrucciones y lógica de procedimientos SQL, los cuales son compilados y almacenados en la misma base. Ya se hizo notar que con estos mecanismos los proveedores de bases de datos introducen la lógica de negocios dentro de su servidor propietario; lo lógico sería mantener una independencia entre datos y códigos.

El rol principal de los procedimientos almacenados es proveer la parte servidora de la lógica de una aplicación cliente/servidor, es decir vendría a reemplazar al servidor de aplicaciones en una arquitectura tres planos (three-tier). Con todo, si bien los procedimientos almacenados permiten a un servidor brindar servicios aptos para OLTP (On Line Transaction Proccesing), no se compara con los rendimientos alcanzados por una arquitectura en tres planos con TP pesado.

Desencadenantes: Son mecanismos que permiten realizar acciones automáticamente sobre los datos, las cuales están asociadas a algún evento definido. Normalmente son implementados a través de procedimientos almacenados. Los eventos a los cuales se hace referencia están asociados a las actualizaciones de tablas mediante sentencias delete, insert o update, y son llamados implícitamente al suceder cualquiera de estos eventos, a diferencia de

los procedimientos almacenados que son llamados explícitamente por un proceso cliente.

Restricciones: Al igual que los desencadenantes, son acciones que se realizan asociadas a algún evento determinado y están orientadas a llevar a cabo validaciones más simples de datos. Los tipos de eventos son los mismos que para los desencadenantes.

Los proveedores de bases de datos optan también por introducir lenguajes de procedimientos para SQL, con lo que se supone que la lógica de negocios (el servidor dentro del modelo cliente/servidor) se haga más exportable, permitiendo migrar la aplicación a cualquier plataforma sobre la cual corre la base de datos. Ejemplos de estos lenguajes de procedimientos para SQL son: PL/SQL (Oracle), Ingres/4GL (Ingres), SPL (Informix), Transac SQL (Sybase).

Servidores de transacciones

Estos tipos de sistemas se pueden implementar con cualquiera de las modalidades cliente/servidor en dos o tres planos, pero incorporan un elemento principal sobre el cual se elabora y basa toda la fortaleza de este modelo, el concepto de transacción.

Tal cual se explica mediante los conceptos de planos, tamaños de componentes y servidores de bases de datos, con un servidor de transacciones el proceso cliente llama a funciones, procedimientos o métodos que residen en el servidor, ya sea

que se trate de un servidor de bases de datos o un servidor de aplicaciones. Lo importante es que el intercambio a través de la red se realiza mediante un único mensaje de solicitud/respuesta, es decir, independientemente de que se necesite ejecutar una o más funciones, una o más instrucciones o sentencias SQL, éstas son agrupadas en una unidad lógica llamada transacción; evitando así el intercambio a través de la red de un mensaje solicitud/respuesta por cada sentencia SQL, el cual es el caso de los sistemas cliente/servidor dos planos, implementados a través de SQL remoto. Estas aplicaciones denominadas OLTP (On Line Transaction Proccesing) están orientadas a dar soporte a los procedimientos y reglas de los sistemas de misión crítica.

7.1.10. Programación Por Capas

La programación por capas es una arquitectura cliente-servidor en el que el objetivo primordial es la separación de la lógica de negocios de la lógica de diseño; un ejemplo básico de esto consiste en separar la capa de datos de la capa de presentación al usuario.[11]

Estructura

La arquitectura de una aplicación es la vista conceptual de la estructura de esta, ya que la aplicación contiene código de presentación (Capa de Presentación), código de procesamiento de datos (Capa de negocio) y código de

[11] http://es.wikipedia.org/wiki/Arquitectura_de_tres_niveles

almacenamiento de datos (Capa de Datos).La arquitectura de las aplicaciones difieren según como está distribuido este código.

El desarrollo se puede llevar a cabo en varios niveles y en caso de algún cambio sólo se ataca al nivel requerido sin tener que revisar entre código mezclado. Permite distribuir el trabajo de creación de una aplicación por niveles.

En una arquitectura de tres niveles, los términos "capas" y "niveles" no significan lo mismo ni son similares, el término "capa" hace referencia a la forma como una solución es segmentada desde el punto de vista lógico: Presentación/ Lógica de Negocio/ Datos.

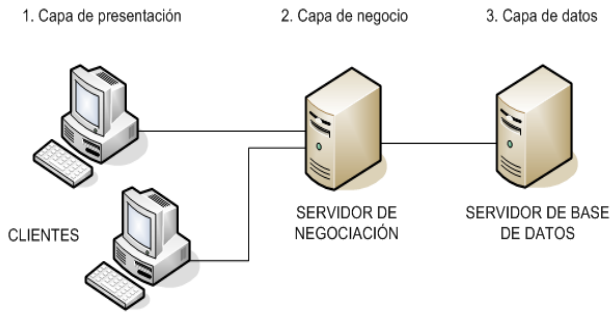
En cambio, el término "nivel", corresponde a la forma en que las capas lógicas se encuentran distribuidas de forma física. Por ejemplo:

- Una solución de tres capas (presentación, lógica, datos) que residen en un solo ordenador (Presentación+lógica+datos). Se dice, que la arquitectura de la solución es de tres capas y *un nivel*.
- Una solución de tres capas (presentación, lógica, datos) que residen en dos ordenadores (presentación+lógica, lógica+datos). Se dice que la arquitectura de la solución es de tres capas y *dos niveles*.
- Una solución de tres capas (presentación, lógica, datos) que residen en tres ordenadores (presentación, lógica, datos). La arquitectura que la define es: solución de tres capas y *tres niveles*.

Funcionamiento

La programación por capas, funciona realizando una separación de la lógica de negocios, de la lógica de diseño, un ejemplo básico de esto es separar la capa de datos de la capa de presentación al usuario.

Ilustración 16. Modelo de programación por capas



La ventaja principal de este estilo, es que el desarrollo se puede llevar a cabo en varios niveles y en caso de algún cambio solo se ataca al nivel requerido sin tener que revisar entre código mezclado. Un buen ejemplo de este método de programación sería: Modelo de interconexión de sistemas abiertos.

Además permite distribuir el trabajo de creación de una aplicación por niveles, de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles; simplemente es necesario conocer la API que existe entre niveles.

Características

La programación por niveles o por capas, se distingue por emplear tres niveles específicos que son:

1. Capa de presentación: Es la que ve el usuario (hay quien la denomina "capa de usuario"), presenta el sistema al usuario, le comunica la información y captura la información del usuario dando un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio. También es conocida como interfaz gráfica y debe tener la característica de ser amigable (entendible y fácil de usar) para el usuario.

2. Capa de negocio: Es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) pues es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación para recibir las solicitudes y presentar los resultados, y con la capa de datos para solicitar al gestor de base de datos almacenar o recuperar datos de él.

3. Capa de datos: Es donde residen los datos y es la encargada de acceder a los datos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Todas estas capas pueden residir en un único ordenador (no es lo típico). Si bien lo más usual es que haya una multitud de ordenadores en donde reside la capa de presentación (son los clientes de la arquitectura cliente/servidor). Las capas de

negocio y de datos pueden residir en el mismo ordenador, y si el crecimiento de las necesidades lo aconseja se pueden separar en dos o más ordenadores. Así, si el tamaño o complejidad de la base de datos aumenta, se puede separar en varios ordenadores los cuales recibirán las peticiones del ordenador en que resida la capa de negocio.

Si por el contrario fuese la complejidad en la capa de negocio lo que obligase a la separación, esta capa de negocio podría residir en uno o más ordenadores que realizarían solicitudes a una única base de datos. En sistemas muy complejos se llega a tener una serie de ordenadores sobre los cuales corre la capa de datos, y otra serie de ordenadores sobre los cuales corre la base de datos.

Estructura

Ilustración 17. Arquitectura Basica

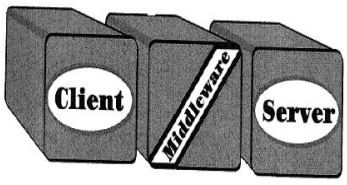
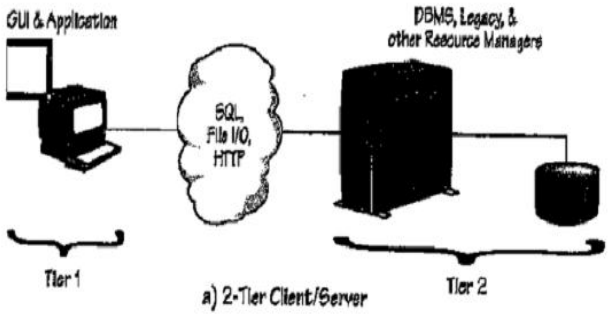


Ilustración 18. Arquitectura cliente servidor dos Capas



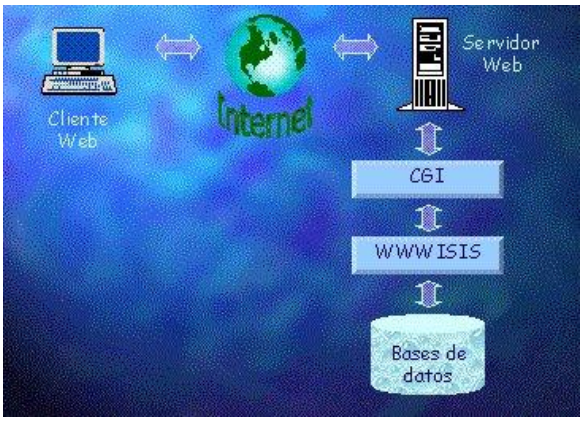


Ilustración 19. Internet

7.1.11. Metodología Rup

PROCESO UNIFICADO DE DESARROLLO DE SOFTWARE (PUD - RUP)

El Proceso Unificado de Desarrollo Software o simplemente Proceso Unificado es un marco de desarrollo de software que se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y por ser iterativo e incremental. El refinamiento más conocido y documentado del Proceso Unificado es el Proceso Unificado de Rational o simplemente RUP.[12]

El Rational Unified Process (RUP) es un proceso de ingeniería de software, creado por Ivar Jacobson, Grady Booch y James Rumbaugh, y cuyo principal objetivo es el de mejorar la productividad y el proceso de desarrollo de software de un equipo de trabajo, así como también dar como resultado la puesta en marcha de las mejores prácticas en el desarrollo de software por parte de los integrantes de dicho equipo, gracias a dichas prácticas, es posible dar cabida dentro del RUP

[12] http://es.wikipedia.org/wiki/Proceso_Unificado

a cualquier tipo de proyectos, incluidos a pequeños proyectos como los de nivel Web.

7.1.11.1. Característica

Flujos De Trabajo:

7.1.11.2. Modelado Del Negocio

Hace referencia a conocer la línea actual de la entidad o negocio, lo cual ayuda en el proceso a establecer las reglas y restricciones que se tienen por parte de la empresa, en este apartado es necesario conocer datos de la empresa tales como misión, visión, objetivo general y objetivos específicos.

7.1.11.3. Requerimientos

En este flujo de trabajo se deben determinar las necesidades funcionales y no funcionales del sistema a desarrollar, así como también la captura de actores y la recolección y depuración de los casos de uso; este paso es de vital importancia ya que los requerimientos permiten establecer la funcionalidad y los insumos necesarios para el proyecto de software.

Requerimientos funcionales: En este apartado se enumeran y describen las operaciones que los posibles usuarios tengan que realizar sobre el sistema, y las operaciones que debe realizar el sistema. El objetivo antes mencionado de este tema se logra gracias al desarrollo del modelo de casos de uso.

Requerimientos No funcionales: Aquí se debe determinar los recursos necesarios para implementar el software, se entiende por recursos el hardware y software necesarios para que el sistema a desarrollar se pueda ejecutar eficientemente.

7.1.11.4. Análisis

Junto con el diseño, el principal objetivo de este flujo de trabajo es trasladar requisitos en especificaciones de implementación, es decir, se debe analizar los requerimientos utilizando el lenguaje de los desarrolladores para producir una vista interna (conceptual) del sistema; en este paso se debe transitar del lenguaje del cliente al lenguaje del desarrollador.

En este apartado se deben realizar las siguientes actividades:

- Realizar un diagrama de secuencia de cada caso de uso.
- Realizar los correspondientes diagramas de actividad.
- Realizar un diagrama de clases de análisis de cada caso de uso.
- Realizar un diagrama de colaboración de cada caso de uso.
- Obtener un diagrama de clases general.

7.1.11.5. Diseño

El objetivo final de este flujo de trabajo es obtener un modelo lógico del sistema a implementar, modelo que debe ser físico y concreto, en pocas palabras, es un

plano de la implementación. Este modelo obtenido debe ser mantenido durante todo el ciclo de vida del software, ya que es el que le da la forma al sistema.

Los pasos a seguir son los siguientes:

- Lo primero que se tiene que realizar es una lista inicial de objetos.
- Se debe refinar las responsabilidades de los objetos.
- Se deben identificar operaciones y atributos de los objetos, e interacciones entre estos.
- Por último se tiene que detallar las relaciones entre los objetos.

NOTA: Los dos anteriores flujos de trabajo deben desarrollarse utilizando UML, lo que implica toda su notación y simbología.

7.1.11.6. Implementación

Esta es la fase en donde se codifican los resultados del diseño en términos de componentes tales como ficheros fuente, ejecutables, scripts, etc. Lo más lógico es que aquí se implementen las clases encontradas durante el diseño convirtiéndolas, si es necesario, en código. Otro de los objetivos de la implementación radica en probar los componentes individualmente e integrarlos en uno o más ejecutables.

Los pasos a seguir son:

- Realizar pruebas de unidad (componentes individuales)

- Unificar componentes de acuerdo al diagrama de componentes, en donde se establece como se organizan los componentes y dependen unos de otros.
- Identificar componentes e incorporarlos al modelo general.

7.1.11.7. Pruebas

El principal objetivo de este flujo es el de realizar pruebas sobre la estructura del sistema que se va formando con los módulos y componentes implementados.
Las pruebas se pueden realizar en diferentes casos:

- Parecidos o Similares, diferenciándose únicamente en algún dato de entrada y/o salida.
- De Instalación en una plataforma: verifican que el sistema puede ser instalado en la plataforma del cliente.
- De Configuración, sobre distintas plataformas: verifican que el sistema funciona correctamente en diferentes configuraciones.
- Negativos, intentando que el sistema falle, por ejemplo utilizando datos no esperados.
- De Tensión o Stress, probando el sistema cuando los recursos son insuficientes o hay competencias por ellos.
- Pueden realizarse pruebas de dos tipos diferentes: de integración y de sistema.

- Pruebas de Integración de Componentes. Se utilizan para verificar que los componentes interaccionan entre sí de un modo apropiado después de haber sido integrados en el sistema. Se toman como Casos de Prueba los casos de uso del diseño. Para ello se utiliza el Diagrama de Secuencia correspondiente y se diseñan combinaciones de entrada y salida del sistema que lleven a distintas utilizaciones de las clases, y en consecuencia de los componentes, que participan en el diagrama.

Realizar las Pruebas de Integración:

- Ejecutar las pruebas.
- Comparar los resultados de las pruebas con los resultados esperados y ver las diferencias.
- Analizar los componentes y las pruebas diseñadas que presentaron esas discrepancias para un posible rediseño del componente o cambios en la prueba.

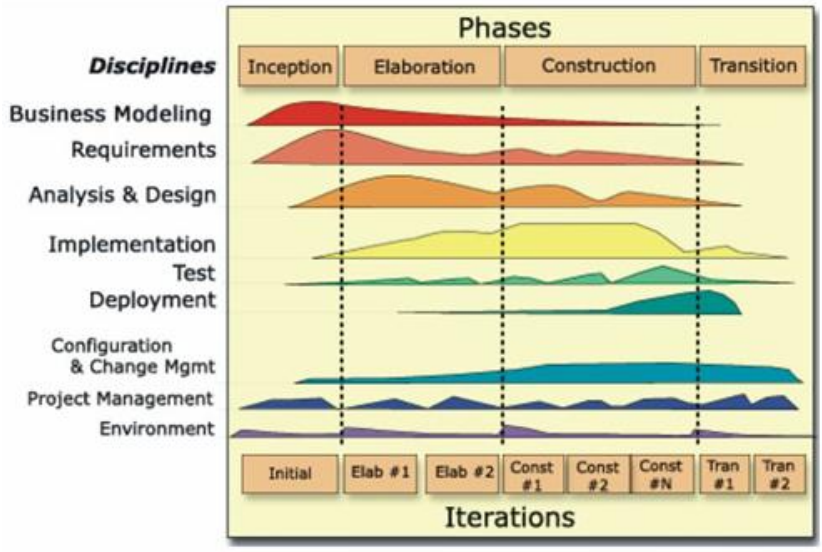
- Diseñar las pruebas del Sistema. Se usa para probar que el sistema funciona globalmente de forma correcta. Cada prueba del sistema prueba combinaciones de casos de uso bajo condiciones diferentes. Se prueba el sistema como un todo probando casos de uso unos detrás de otros y, si es posible, en paralelo. Se trata de ver que cada caso de uso funciona adecuadamente en distintas configuraciones hardware, de carga, con varios actores a la vez, en distinto orden, etc.

Realizar la Prueba del Sistema:

- La prueba del sistema puede empezar cuando las pruebas de integración indican que el sistema satisface los requisitos de calidad fijados durante las pruebas. Por ejemplo, el 90% de las pruebas de integración se realizan con el resultado esperado.
- La prueba del sistema se lleva a cabo de forma similar que las pruebas de integración.
- Los diseñadores de las pruebas evalúan los resultados de la prueba, fundamentalmente a partir de dos métricas:
- Completitud de la prueba: indica el porcentaje de casos de prueba que han sido ejecutados correctamente y el porcentaje de código que ha sido probado.
- Fiabilidad: se basa en el análisis de la tendencia en los errores detectados y de los resultados esperados. Lo usual es que el número de errores se incremente rápidamente al inicio de las pruebas, se mantenga estable posteriormente durante un tiempo y finalmente empiece a decrecer. Basándose en el análisis de la tendencia de los errores detectados se puede sugerir realizar pruebas adicionales, relajar el criterio de pruebas si los objetivos de calidad se pusieron muy altos, o revisar la parte del sistema que no cumple los requisitos de calidad.[13]

[13] Tomado de proyecto Unidad didáctica computacional UDIC basada en un simulador para el proceso de destilación simple y fraccionada en el contexto de la enseñanza/aprendizaje de la química.

Ilustración 20. Fases del modelo RUP



7.1.12. Lenguaje De Programación

7.1.12.1. Java

El tipo de lenguaje que usaremos es el de Java que es es un lenguaje de programación de alto nivel orientado a objetos, desarrollado por James Gosling en 1995. El lenguaje en sí mismo toma mucha de su sintaxis de Visual Basic, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. La memoria es gestionada mediante un recolector de basura.

Las aplicaciones Java están típicamente compiladas en un bytecode, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo

para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible.

La implementación original y de referencia del compilador, la máquina virtual y las bibliotecas de clases de Java fueron desarrolladas por Sun Microsystems en 1995. Desde entonces, Sun ha controlado las especificaciones, el desarrollo y evolución del lenguaje a través del Java Community Process, si bien otros han desarrollado también implementaciones alternativas de estas tecnologías de Sun, algunas incluso bajo licencias de software libre.

Entre diciembre de 2006 y mayo de 2007, Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL, de acuerdo con las especificaciones del Java Community Process, de tal forma que prácticamente todo el Java de Sun es ahora software libre aunque la biblioteca de clases de paginas web comprendidas en las librerias de objetacion de objetos para ser compilados como aplicaciones comprimidas no estan totalmente acopladas de acuerdo con Sun que dice que se requiere un interprete para ejecutar los programas de Java.[14]

[14] http://es.wikipedia.org/wiki/Java_%28lenguaje_de_programaci%C3%B3n%29

7.1.13. Estado Del Arte

Gracias a que la popularidad en los dispositivos móviles ha incrementado grandemente no solo por su posibilidad de comunicación sino por el contrario debido a que permiten una gran variedad de utilidades como cámara digital, reproductor de música, entre otros.

Así una labor de los ingenieros en los años recientes ha sido en crear aplicaciones donde los usuarios puedan "consumir" desde los celulares, luego la computación móvil va con miras en el desarrollo de todo lo relación con dispositivos móviles, donde no solo incluye los celulares como tal sino PDAs y demás dispositivos que puedan tener un desplazamiento móvil.[15]

Este tipo de computación, por así decirlo, tiene 4 restricciones que marcan una gran diferencia entre un computador portátil, por ejemplo, que es móvil y un celular.

- Pobreza en los recursos.
- Sensibles a los daños.
- Conectividad altamente baja.
- Finita fuente de energía.

Pero entonces, por qué los dispositivos móviles han tenido una gran acogida en el público en general, básicamente por la personalización, localización y la posibilidad de estar siempre conectado han hecho de este una gran fuerza de

[15] http://eprints.rclis.org/bitstream/10760/15898/1/informeapeimovilidad.pdf

mercado. Personalización, permite al usuario expresar sus ideas y gustos mostrándose de una forma libre y sin prejuicios debido a que, por lo general, es de uso personal e individual. Localización, A diferencia de las computadoras convencionales un dispositivo móvil, le permite conocer la posición del usuario en tiempo real.[16]

Ahora bien, los dispositivos móviles que están "entrando" al mercado ya han solucionado muchos de los inconvenientes anteriormente mencionados pues estos poseen una mayor capacidad de computo, una mayor duración de batería. Aun estos poseen tecnologías como Wifi, Bluetooth, GPS y demás convirtiéndoles en dispositivos más ricos para el usuario.[17]

Algunas de las investigaciones que se han adelantado sobre dispositivos móviles son las siguientes:

1. El exalumno César Esteban Castañeda Ruíz, de la facultad de ingeniería de sistemas de la Pontificia Universidad Javeriana en el año 2009, elaboró un proyecto llamado Middleware para la definición e implementación de servicios genéricos independientes de la plataforma orientado a dispositivos móviles. Según el autor, este trabajo de grado fue diseñado con el fin de crear un Middleware

[16] http://www.slideshare.net/mon08/evolucion-de-las-tecnologias-en-los-dispositivos-moviles-1544261

[17] Cobo Romaní, Cr; Pardo Kuklinski, H (2007). Planeta Web 2.0 : Inteligencia Colectiva o Medios Fast Food. Cap.6

independiente de la plataforma que permitiera desarrollar, definir e implementar servicios genéricos de forma estándar, al mismo tiempo que mantuviera brevedad y sencillez en consideración con los recursos limitados que existían para ese entonces en los dispositivos móviles. Pese a que César Castañeda relaciona y explica las diferentes tecnologías que existen, lo que realmente busca con su investigación es eliminar el problema que presenta el desarrollador en cuanto a la interoperabilidad existente entre plataformas y software y a su vez facilitar la tarea de definición e implementación de servicios en dispositivos móviles enfocándose en el estudio de las tecnologías de transferencia inalámbrica de datos disponibles como Wifi, Wimax, GSM, GPRS, EDGE, UMTS, HSDPA y en los protocolos que se han utilizado para la implementar exitosamente servicios tales como: SMS, MMS, WAP, TCP/IP y UDP/IP. Finalmente César concluye su trabajo manifestando que el diseño de servicios bajo el esquema de interfaces remotas de cliente y servidor cubre el objetivo de lograr la interoperabilidad, además de generalización para servicios que no necesariamente son LBS (Location Based Services).

Por otra parte se evidencia que en este trabajo de grado, a pesar de que se nombran varios conceptos y elementos que fueron necesarios para nuestro proyecto, realmente no interviene en la ejecución de un aplicativo que desarrolle ningún tipo de solución matemática y mucho menos su graficación.[18]

[18] http://www.cnc.gov.ar/ciudadanos/telefonia_movil/evolucion.asp#iconsumo

2. Otro de los proyectos que se han desarrollado en este tema es el proyecto, que fue desarrollado en por Michal Yerushalmy, quien es profesora de la enseñanza de las matemáticas en el departamento de la facultad de la educación, la señora Michal es la directora del centro de investigación del instituto de alternativas en educación en la universidad de Haifa, Israel En la publicación de la página de internet (http://www.edu.haifa.ac.il/personal/michalyr/bio.html), se menciona que los intereses de esta docente es centrar el aprendizaje especialmente de las matemáticas a través de entornos informatizados, adicionalmente el enfoque de la investigación de Yerushalmy fue escribir y diseñar números paquetes de software de geometría, algebra y calculo y desarrolló el interactivo “Visualmath” que es el plan de estudios de la escuela secundaria web matemáticas. Su proyecto se centra en el aprendizaje con diagramas interactivos en libros electrónicos interactivos y en mLearning, además diseñó el Math4mobile el cual ofrece maneras de hacer que la tecnología esté disponible para la construcción de conocimiento para todos en todas partes. A continuación detallaremos algunos de los proyectos actuales del instituto de alternativas en la educación en el que participó la señora Michal, además analizaremos los beneficios que ofrecen estos programas, la diferencia existente entre ellos mismos y finalmente el plus adicional que con nuestro proyecto ofrece a partir de

7.1.13.1. Math4mobile

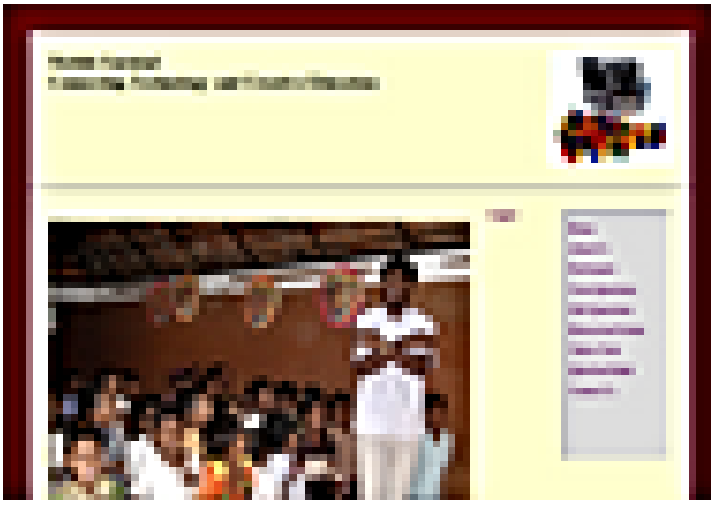
El proyecto MathBook Interactivo y el VisualMath: Function, examina las matemáticas de aprendizaje con diagramas interactivos, en los libros de texto de matemáticas interactivas; este tiene por objeto identificar las prácticas relacionadas con el diseño del libro de texto electrónico interactivo y centrarse en las funciones de los diagramas interactivos, teniendo en cuenta que en el instructivo se aclara que un diagrama interactivo es una aplicación de software relativamente pequeño y simple (un applet) entorno a un ejemplo pre-constructivo visual. El instituto está interesado en el estudio de problemas de los estudiantes para resolver rutinas utilizando diagramas interactivos. Micro-análisis del discurso y los gestos de un grupo de estudiantes de 13-14 años de edad que trabajan álgebra con unos diagramas interactivos que les permiten determinar cómo los componentes diseñados de los diagramas interactivos se instrumentan para participar en los procesos de resolución de problemas. (Traído de (https://sites.google.com/site/interactivediagrams/Home).

Ilustración 21. Ambiente de aplicativo Math4mobile

7.1.13.2. Mobile Gurukul

Propone utilizar la capacidad de penetración de los teléfonos móviles y, principalmente, para permitir un sistema de tutoría a distancia para maestros de escuela. Mediante la combinación de los teléfonos individuales con tecnología web móvil Gurukul trata de ayudar a superar la brecha digital y para llenar el vacío creado en las zonas rurales debido a la falta de equipos de poder. Trata de llenar el aislamiento de los educadores en las aldeas, que a menudo funcionan como un solo maestro en una escuela primaria o como profesor de la asignatura única en una escuela secundaria. El instituto se considera como una comunidad abierta que considera que la motivación, las ideas innovadoras y el liderazgo a largo plazo son las cualidades que más avanzarían educación rural. (www.mobilegurukul.org).

Ilustración 22. Ambiente de aplicativo Mobile Gurukul



7.1.13.3. Clickable

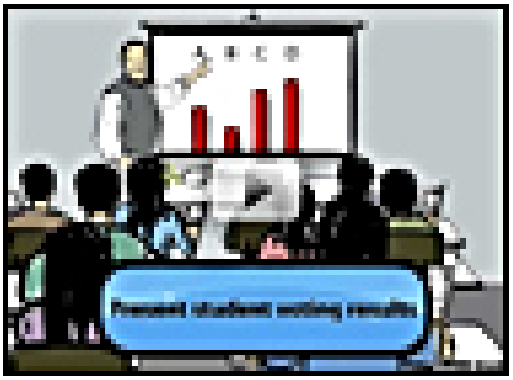
Utiliza la tecnología de código de barras 2D para convertir diagramas estáticos, impresos y ejercicios en las actividades interactivas dirigidas por aplicación Java. Esta innovación es impulsada por la suposición de que los textos impresos son valiosas y fácil de usar, pero debe ser reforzada por las actividades interactivas. (Arik Weizman - M.Sc. proyecto en el departamento de Ciencias de la Computación de la Universidad de Haifa y con la ayuda de 3GVision (2008)).

Ilustración 23. Diseño de ambiente de aplicativo Clickable

7.1.13.4. Click2go

El sistema de interacción Mobile consiste en apoyar a los maestros en la organización y la evaluación de la construcción del conocimiento en grandes grupos. Click2go es un sitio web que ofrece a los profesores un sistema cómodo y fácil de comunicarse con los estudiantes que están registrados para el curso específico a través de sus teléfonos móviles personales. (Desarrollado por Ron Shahanovsky y Gal Star - B.Sc final del proyecto (2009)).

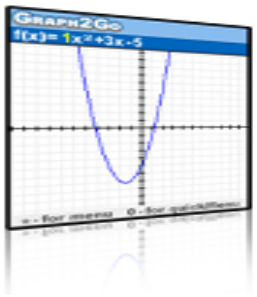
Ilustración 24. Ambiente de aplicativo Click2go



7.1.13.5. Graph2Go

Es una calculadora graficadora de propósitos específicos que opera para un conjunto de expresiones de Funciones.

Ilustración 25. Ambiente de aplicativo Graph2Go



Características

Son características básicas de Graph2Go las vinculadas a su posibilidad de:

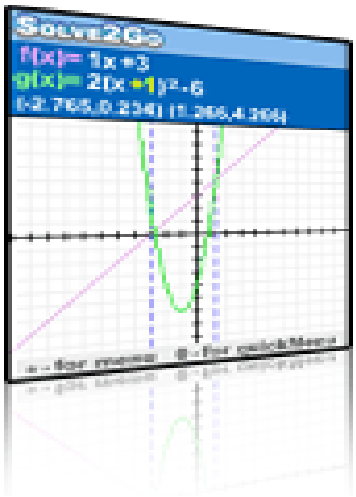
- Graficar funciones en relación con expresiones de una variable.
- Trazar dinámicamente el gráfico de expresiones en la medida que se las transforma.

- Señalar los puntos correspondientes a máximos, mínimos y puntos de inflexión y exponer sus valores.
- Representar gráficamente una expresión y su función derivada.
- Trazar gráfica y simbólicamente la familia resultante de integrar una función.
- Expresar el área correspondiente a una integral definida de una función.
- Escalar y acercar con efectos de zoom los gráficos.

7.1.13.6. Solve2Go

Respalda la resolución de ecuaciones e inecuaciones en tanto habilita el establecimiento de presunciones que pueden emerger de la inspección visual de lo representado. Cualquier conjetura elaborada puede refutarse o ratificarse apelando a ejemplos que el recurso despliega y que pueden relacionarse con las anotaciones de operación simbólica.

Ilustración 26. Modelo de aplicativo Solve2Go



Características

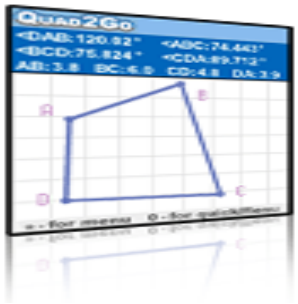
Pueden especificarse dos expresiones de sendas funciones seleccionándolas de una lista dada de expresiones paramétricas de funciones. Solve2Go selecciona valores numéricos aleatorios para los parámetros y grafica las dos funciones.

También marca los puntos de intersección cuando existen y están visibles en pantalla. Para explorar las soluciones de otras ecuaciones o inecuaciones del mismo tipo (seleccionado, es recomendable usar el cambio interactivo de constantes y coeficientes en cada expresión, transformar los gráficos y ver si cambian las soluciones y cómo. El diseño de Solve2Go se basa en las características especiales de Graph2Go.

7.1.13.7. Quad2Go

Es un recurso portable que ofrece medios para aprender sobre cuadriláteros, produciendo ejemplos, observando y experimentando para avanzar desde los intentos de cubrir ciertos propósitos de un modo particular a la formulación más general, de conjeturas.

Ilustración 27. Modelo de aplicativo Quad2Go



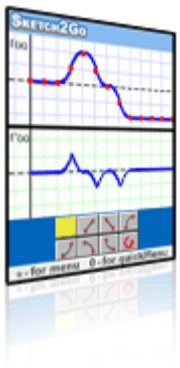
Características

Quad2Go es un recurso portátil que pone a mano, medios para aprender sobre cuadriláteros, produciendo ejemplos, observando y experimentando para avanzar desde la exploración con propósitos particulares a la formulación más general, de conjeturas. Como otros utilitarios (SketchPad Geométrico, Conjeturador- Supposer Geométrico, GeoGebra o Cabri), Quad2Go ofrece un sistema para el tratamiento dinámico de objetos geométricos, útiles para manejarlos, recursos de maniobra e instrumentos de medida. Limitado a cuadriláteros y al marcado de sus diagonales.Quad2Go permite construir, observar y transformar todos los que pueden trazarse con regla y compás, medir lados, ángulos y áreas, y transformar la construcción desplazando, arrastrando y transformando la figura. La figura se modifica al arrastrar alguno de sus elementos que se traslada sobre la pantalla. El contexto dinámico admite el tratamiento experimental de objetos geométricos acorde al propósito de quien simultáneamente indaga en torno a sus propias acciones y a los resultados, para relacionarlos en un aprendizaje constructivo que parte del reconocimiento causal hacia una lógica de significaciones.

7.1.13.8. Sketch2Go

Es una herramienta cualitativa de graficación. Los gráficos se trazan empleando siete iconos representativos de funciones constantes, crecientes y decrecientes que cambian a razones constantes, crecientes y decrecientes.

Ilustración 28. Aplicativo Sketch2Go



Características

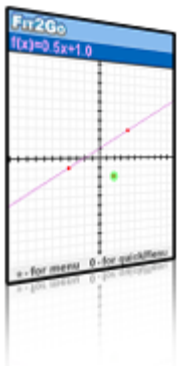
Sketch2Go es una herramienta cualitativa de graficación. Los gráficos se trazan empleando siete iconos representativos de funciones constantes, crecientes y decrecientes que cambian a razones constantes, crecientes y decrecientes. Se basa en el original R&D desarrollado por Schwartz y Yerushalmy (1995) y Shternberg & Yerushalmy (2001), que proponían un puente intermediando la representación basada en la función y su vocabulario. Los siete íconos gráficos describen el cambio tanto en la función como en su razón de cambio. Sketch2Go es una versión del Qualitative Derivative Grapher programado por Alexander Zilber para el CET (Centro de Educación Tecnológica). La modelización matemática no puede llevarse a cabo completamente con este sistema cualitativo de signos de funciones constantes, crecientes y decrecientes. Pero el conjunto de los siete íconos respalda el formato de la construcción matemática a través de un lenguaje desarrollado para lidiar con escenarios físicos, ayudando a establecer los fundamentos de la enseñanza de pre-cálculo y cálculo. Sketch2Gorespalda la s abstracción de fenómenos cotidianos usando un

reducido juego de signos matemáticos que pueden manipularse en la pantalla como objetos semi-concretos.

7.1.13.9. Fit2Go

Es una herramienta para la graficación de funciones lineales y cuadráticas y ajuste de curvas. Permite apreciar un fenómeno, identificar variables, llevar adelante experimentos y tomar medidas para construir modelos de un fenómeno.

Ilustración 29. Aplicativo Fit2Go



Características

Fit2Go es una herramienta para la graficación de funciones lineales y cuadráticas y ajuste de curvas. Permite apreciar un fenómeno, identificar variables, llevar adelante experimentos y tomar medidas para construir modelos de un fenómeno. Fit2Go es una herramienta para la graficación de funciones lineales y cuadráticas y ajuste de curvas. Permite apreciar un fenómeno, identificar variables, llevar adelante experimentos y tomar medidas para construir modelos de un fenómeno. Fit2Go facilita el ingreso de datos y el control dinámico en una inspección visual de los puntos y sus valores. Tras seleccionar el tipo de modelo, decisión clave que protagoniza activamente quien interpreta apelando a la herramienta (lo que superar el librarla a un ajuste automático), Fit2Go presenta

una recta específica (si se marcan sólo dos puntos) o una función cuadrática (si están marcados tres). Los casos interesantes aparecen cuando se cuenta con pocos o demasiados puntos. Fit2Go no se ajusta al modelo por interpolación cruzando por todos los puntos sino que aleatoriamente traza curvas optativas para cruzar por subconjuntos de puntos y deja al activo intérprete a cargo de considerar las alternativas. Cuando las condiciones son escasas, en contrapartida,Fit2Go ofrece una familia de gráficos de entre los cuales es posible ir decidiendo.

CONCLUSIONES

Los dispositivos móviles son una herramienta clave en la cotidianidad y la vida laboral y estudiantil; debido a los avances de hardware y software, estos dispositivos que en su comienzo servían solo como terminales de voz, hoy se fortalecen al ser capaces de ejecutar diferentes tipos de aplicaciones.

Las aplicaciones *Online* tienen la ventaja de estar disponibles para todos los dispositivos que cuenten con un navegador Web sin necesidad de ningún proceso de instalación adicional. También al ser una aplicación centralizada cualquier mantenimiento, actualización o corrección de algún error en la aplicación, se ve inmediatamente reflejado para todos los usuarios.

La desventaja de las aplicaciones *Online*, consiste en que éstas se encargan de ejecutar la lógica de la aplicación y de manejar el interfaz de usuario; en resumen casi toda la carga de trabajo es ejecutada en el servidor Web.

Si los usuarios simultáneos crecen también crece el trabajo en el servidor, por tanto aumenta su tiempo de respuesta y disminuye su eficiencia. En resumen puede atender a menos usuarios simultáneos que una aplicación *Smart Client*.

Las aplicaciones *Smart Client* o cliente inteligente ejecutan parte de la carga de trabajo, y exclusivamente se conectan al servidor para obtener datos. Al

encargarse del manejo de la interfaz de usuario aliviana el trabajo del servidor; por consiguiente éste puede atender a más usuarios y con tiempos de respuesta adecuados.

Actualmente ya existe una amplia oferta de dispositivos móviles en el mercado y los fabricantes distribuyen nuevos modelos constantemente. Como resultado de esta gran variedad de dispositivos, se deben afrontar los problemas cada vez que se desarrolla una aplicación Web móvil.

Las soluciones móviles están mostrando sus beneficios para la gestión de las empresas en la mejora de la productividad, en la creación de nuevos servicios.

Desarrollo de Aplicaciones móviles permiten estar a la vanguardia en comunicaciones y sistemas de información.

Los teléfonos celulares, así como todos los dispositivos móviles, son aparatos que en la actualidad utiliza la mayoría de la gente. Al igual que las computadoras, estos artefactos han dejado de ser un lujo, pues se han convertido en una necesidad. Además de cumplir con la función básica de un celular (realizar y recibir llamadas telefónicas), la mayoría de ellos cuentan con funciones especiales: el envío de mensajes de texto cortos (SMS), mensajes de texto multimedia (MMS), mensajes instantáneos (IM), correos electrónicos, navegar en

Internet y administrar información personal (PIM) con ayuda de ciertas aplicaciones, entre otras.

Casi todos los celulares permiten a los usuarios la instalación de ciertas aplicaciones, así como el almacenamiento de información personal y confidencial, sin importar el sistema operativo, la forma en la que se sincroniza la información o cómo se conectan con los equipos de cómputo. Los celulares son parte de la tecnología que nos rodea hoy en día. Su mejora y actualización es constante.

Cuando un celular es involucrado en un crimen o en un incidente, los analistas forenses requieren de herramientas que permitan obtener una apropiada y rápida recuperación de la información almacenada en el dispositivo. La información obtenida, después de ser analizada, servirá para redactar un reporte detallado de las actividades realizadas, incluyendo fechas, con la finalidad de buscar evidencias que revelen la causa y forma en la que se llevó acabo un delito o se violó una política, en algunos casos esta información puede obtenerse, aun cuando ésta haya sido borrada.

RECOMENDACIONES

Es recomendable diseñar un sistema con una estructura de capas funcionales. Se debe separar en capas las funcionalidades de acceso a datos, lógica de negocio e interfaz de usuario. El organizar de esta manera el código, permite agilizar los procesos de mantenimiento de software. Además es posible, como en el presente Proyecto de titulación, hacer diferentes implementaciones de la capa de interfaz de usuario; las aplicaciones *Online* y *Smart Client* son implementaciones de esta capa.

Se recomienda utilizar procedimientos almacenados para el acceso a datos, ya que éstos son pre-compilados y se ejecutan con mayor rapidez, permitiendo obtener los datos de una manera eficiente.

Es recomendable utilizar un emulador de dispositivo en la fase de desarrollo de software, pero para probar el funcionamiento de la versión final es aconsejable realizarlo en un dispositivo real.

Antes que una aplicación Web salga a un ambiente de producción, ésta debe pasar un proceso de pruebas de rendimiento, que ayude a establecer los límites de esta aplicación o encontrar “cuellos de botella” que afecten el desempeño de la aplicación Web. El que una aplicación Web falle es desastroso, pues afecta al

número de usuarios conectados simultáneamente y dependiendo los servicios que preste puede causar un malestar
masivo en los usuarios finales.

Para realizar las pruebas se deben considerar períodos de prueba mayores a 20 minutos y asegurarse que los cliente que generen tráfico tengan un porcentaje de procesamiento menor al 75%.

Es recomendable que aplicaciones de uso masivo con una funcionalidad bien definida, donde el desempeño es muy importante, sean desarrolladas como 199 aplicaciones inteligentes o *Smart Client*; un claro ejemplo de una aplicación *Smart Client* es *Microsoft Messenger*. Mientras que aplicaciones de comercio electrónico, en donde van a existir cambios constantes de catálogos, precios, etc., que pueden ser fácilmente manejados en un servidor centralizado, se deben pensar como aplicaciones *Online*.

Se recomienda que los servicios en línea, que en la actualidad se prestan para PCs, migren para ser accesibles a dispositivos móviles, pues debido al masivo uso de móviles estos servicios serán muy solicitados.

Para que ocurra esto la conectividad de datos para los dispositivos móviles deberá tener costos atractivos
para el usuario final.

BIBLIOGRAFIA

BOEHM, Barry. Characteristic of software quality. TRW Thompson Ramo Wooldridge, Inc, 1978.

BROCKBANK, Bray J. Learning Management System for e-learning, in The AMA handbook of e-learning: Effective design, implementation, Provo 2004.

JACOBSON, Ivar. El proceso unificado de desarrollo de software. Editorial Addison Wesley, 2001.

NEBENDAHL, Dieter. Sistemas Expertos, Siemens, 1998

NELSEN Jacob. Usabilidad Diseño de sitios Web. Prentice Hall, 2000.

RUSSEL, Stuart. Inteligencia Artificial, Prentice Hall. 1996.

CASTELLS, M., FERNÁNDEZ Ardévol, LINCHUAN Qiu, J. y Sy, A. Comunicación móvil sociedad, 2007.

BOROTIS, S. y POULYMENAKOU, A. E-Learning Componentes: aspectos clave a tener en cuenta antes de adoptar Aprendizaje e-Intervenciones. Proc. De eLearn, Washington, DC. 2004.

INFOGRAFIA

http://www.utpl.edu.ec/blog/zamora/2008/02/13/arquitectura-por-capas/

http://es.wikipedia.org/wiki/Arquitectura_de_tres_niveles

http://es.wikipedia.org/wiki/Arquitectura_de_software

http://es.wikipedia.org/wiki/Arquitectura_orientada_a_servicios

http://209.85.135.104/translate_c?hl=es&u=http://www.aadl.info/&prev=/search%3Fq%3DADLs%26hl%3Des

http://209.85.135.104/translate_c?hl=es&u=http://www.cs.cmu.edu/~acme/&prev=/search%3Fq%3DADLs%26hl%3Des

http://translate.google.com/translate?hl=es&sl=en&u=http://128.194.106.6/~baum/linuxlist/linuxlist/node7.html&sa=X&oi=translate&resnum=2&ct=result&prev=/search%3Fq%3DADL%2B-%2BAESOP%26hl%3Des%26sa%3DG

http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/lenguaje.mspx#ECHAE